Università di Genova

DIPARTIMENTO DI
INFORMATICA, BIOINGEGNERIA,
ROBOTICA E INGEGNERIA DEI SISTEMI

Quantum inspired approach for early classification of time series

Alberto Cabri

Theses Series DIBRIS-TH-2019-21

Università di **Genova**

Dipartimento di Informatica, Bioingegneria,
Robotica ed Ingegneria dei Sistemi

Ph.D. Thesis in
Computer Science and Systems Engineering
Computer Science Curriculum

Quantum inspired approach for early classification of time series

by

Alberto Cabri

December, 2019

Ph.D. Thesis in Computer Science and Systems Engineering (S.S.D. INF/01)
Dipartimento di Informatica, Bioingegneria,
Robotica ed Ingegneria dei Sistemi
Università di Genova

Candidate
Alberto Cabri
alberto.cabri@dibris.unige.it

Title
Quantum inspired approach for early classification of time series

Advisors
Francesco Masulli
DIBRIS, Università di Genova
francesco.masulli@unige.it

Stefano Rovetta
DIBRIS, Università di Genova
stefano.rovetta@unige.it

External Reviewers
Jacob Biamonte
Skolkovo Institute of Science and Technology (RU)
jacob.biamonte@qubit.org

Daniel Neagu
University of Bradford (UK)
D.Neagu@bradford.ac.uk

Domenico Tegolo
Università di Palermo
domenico.tegolo@unipa.it

Location
DIBRIS, Univ. di Genova
Via Opera Pia, 13
I-16145 Genova, Italy

Submitted On
December 2019

To my beloved family,
who supported me throughout this challenging experience.

Abstract

Is it possible to apply some fundamental principles of quantum-computing to time series classification algorithms?

This is the initial spark that became the research question I decided to chase at the very beginning of my PhD studies. The idea came accidentally after reading a note on the ability of *entanglement* to express the correlation between two particles, even far away from each other.

The test problem was also at hand because I was investigating on possible algorithms for real time bot detection, a challenging problem at present day, by means of statistical approaches for sequential classification.

The quantum inspired algorithm presented in this thesis stemmed as an evolution of the statistical method mentioned above: it is a novel approach to address binary and multinomial classification of an incoming data stream, inspired by the principles of Quantum Computing, in order to ensure the shortest decision time with high accuracy.

The proposed approach exploits the analogy between the intrinsic correlation of two or more particles and the dependence of each item in a data stream with the preceding ones.

Starting from the *a-posteriori* probability of each item to belong to a particular class, we can assign a Qubit state representing a combination of the aforesaid probabilities for all available observations of the time series. By leveraging *superposition* and *entanglement* on subsequences of growing length, it is possible to devise a measure of membership to each class, thus enabling the system to take a reliable decision when a sufficient level of confidence is met.

In order to provide an extensive and thorough analysis of the problem, a well-fitting approach, found in literature, [23] for bot detection was replicated on our dataset and later compared with the statistical algorithm to determine the best option. The winner was subsequently examined against the new quantum-inspired proposal, showing the superior capability of the latter in both binary and multinomial classification of data streams.

The validation of quantum-inspired approach in a synthetically generated use case, completes the research framework and opens new perspectives in on-the-fly time series classification, that we have just started to explore.

Just to name a few ones, the algorithm is currently being tested with encouraging results in predictive maintenance and prognostics for automotive, in collaboration with University of Bradford (UK), and in action recognition from video streams.

Publications

Some ideas and figures may have appeared previously in the following publications:

[1] Amr Abdullatif, Francesco Masulli, Stefano Rovetta and Alberto Cabri. 'Graded possibilistic clustering of non-stationary data streams'. In: International Workshop on Fuzzy Logic and Applications. Springer. 2016, pp. 139–150.

[2] A. Cabri, F. Masulli and S. Rovetta. A quantum-inspired classifier for early decision making. Submitted to a Journal. Aug. 2019.

[3] A. Cabri, G. Suchacka, S. Rovetta and F. Masulli. 'Online Web Bot Detection Using a Sequential Classification Approach'. In: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). June 2018, pp. 1536–1540.

[4] T Valls Mataró, Francesco Masulli, Stefano Rovetta, Alberto Cabri, C Traverso, E Capris and S Torretta. 'An assistive mobile system supporting blind and visual impaired people when are outdoor'. In: 2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI). IEEE. 2017, pp. 1–6.

[5] Stefano Rovetta, Alberto Cabri, Francesco Masulli and Grażyna Suchacka. 'Bot or Not? A Case Study on Bot Recognition from Web Session Logs'. In: Italian Workshop on Neural Nets. Springer. 2017, pp. 197–206.

[6] Stefano Rovetta, Francesco Masulli and Alberto Cabri. 'Measuring clustering model complexity'. In: International Conference on Artificial Neural Networks. Springer. 2017, pp. 434–441.

[7] Stefano Rovetta, Francesco Masulli and Alberto Cabri. The "Probabilistic Rand Index": A look from some different perspectives. Italian Workshop on Neural Nets. in press. Springer.

[8] Stefano Rovetta, Zied Mnasri, Francesco Masulli and Alberto Cabri. 'Emotion recognition from speech signal using fuzzy clustering'. In: 2019 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology (EUSFLAT 2019). Atlantis Press, Aug. 2019. ISBN: 978-94-6252-770-6. DOI: https://doi.org/10.2991/eusflat-19.2019.19.

[9] G. Suchacka, A. Cabri, S. Rovetta and F. Masulli. Efficient On-the-fly Web Bot Detection. IEEE Transactions on Knowledge and Data Engineering. Under review. 2019.

Acknowledgements

Many thanks go to Professor Francesco Masulli and Professor Stefano Rovetta for their continous support and mentoring, to Professors Grażyna Suchacka, Daniel Neagu and Felician Campean for offering the opportunity of testing the proposed algorithms on real world data streams.

Additional thanks go to Professor Paolo Solinas for reviewing the Quantum Computing theoretical background.

Contents

List of Figures

List of Tables

Acronyms

1NN 1-Nearest Neighbor

ADS Average Decision Sequence

ANN Artificial Neural Network

ARMA Auto Regressive Moving Average

ARIMA Auto Regressive Integrated Moving Average

ART2 Adaptive Resonance Theory 2

CAPTCHA Completely Automated Public Turing test to tell Computers and Humans Apart

CBDTW Correlation Based Dynamic Time Warping

CSV Comma Separated Values

DBSCAN Density-Based Spatial Clustering of Applications with Noise

DDoS Distributed Denial of Service

DTMC Discrete Time Markov Chain

DTW Dynamic Time Warping

ECDIRE Early Classification framework based on class Discriminativeness and Reliability of Predictions

ECM Early Classification Model

GPCM Graded Possibilistic c-Means

HMM Hidden Markov Model

HTTP Hyper-Text Transfer Protocol

IID independent identically distributed

IoT Internet of Things

kNN k-Nearest Neighbor

LAMP Linux, Apache, MySQL, PHP

MCL Markov clustering

MLP Multi-Layer Perceptron

MSD Multivariate Shapelets Detection

MTS Multivariate Time Series

OTF on-the-fly

PCA Principal Component Analysis

PSO Particle Swarm Optimization

QEMC Quantum-inspired Entangled Multinomial Classifier

SOA state-of-the-art

SOM Self-Organizing Maps

SPRT Sequential Probability Ratio Test

SVM Support Vector Machine

URI Universal Resource Identifier

UTS Univariate Time Series

PART I

Background and Proposed Approach

The first part of this thesis introduces the details of the research question that the proposed quantum-inspired algorithm aims to answer, analyzing in depth the most relevant state of the art approaches. Two different early decision approaches have been developed in my research activity: the first is a Bayes optimal approach, proposed by Wald in 1945 with the name of Sequential Probability Ratio Test, whereas the second stems from a personal proposal that the entanglement property of quantum mechanics could somehow express the intrinsic nature of temporal sequences. The theoretical foundation of the novel quantum-inspired algorithm is described both in binary and multinomial settings.

1

Introduction

In the era of Big Data, huge amounts of data are collected at high velocity in several industrial contexts, raising new challenges for data analysts and researchers, with particular regard to timely recognition of anomalous or critical behaviors, that can be detected only through continuous monitoring of incoming data and proper decision making.

Whenever the collected data samples are indexed on time, the relevant dataset represents a time series where each data sample is somehow related to its neighbors: being able to automatically classify a sequence is a highly valuable task and even more important is the ability to label a time series with the fewest possible observations [66, 88].

There are areas of application for time series in which classification accuracy is the essential point and no particular attention is given to the rapidity of decision.

For instance, forgeries detection on signatures is an interesting application that does not require on-the-fly classification but the highest possible accuracy is the key performance metric. Conversely, timely decisions are requested on an extrusion line in order to spot and fix possible defects before the product integrity gets compromised.

These simple considerations puts in evidence the duality of time series analysis, which is reflected in the approaches that must be selected to tackle the various problems. As pointed out in [65], the approaches can be divided in *offline*, when a complete sequence is considered before labeling, or *online* (alias on-the-fly), if the decision has to be taken as soon as possible on the base of incoming measures.

The latter approach is commonly termed *early classification* of time series. Examples of such challenging problems can be found in several industrial contexts, as shown in Table 1, and they are often related to the availability of new sophisticated equipment or IoT sensors which enable harvesting huge amounts of data, most frequently as a sequence of correlated events or measures.

Even a video source, despite more specific and effective techniques are available based on convolutional neural networks [10] architectures, could ideally be treated as a sequence of time related events, each event being the video frame.

Table 1: Example of early classification problems for time series

Task	*Description*
cyber-security	on threat detection; being able to timely discover undesired access to web sites and circumvent misuse of network resources can prevent fraudulent activities against service providers and the resulting economical and trust loss;
disease prevention	early recognition of a disease onset not only can save or extend patients' lives but also can guarantee a better after treatment course and limit the costs for medical care when it allows for delaying chronic pathologies;
seizure alert	monitoring some physiological parameters, such as oxygen saturation or tachycardia, in hospitalized patients may assist caregivers in prompt recognition of physical deterioration by raising preventative alerts;
predictive maintenance	identification of unusual patterns in the behavior of an industrial system can reduce both the downtime and the maintenance costs, especially when the breakdown of a component affects many dependent elements of the system. Moving from a preventive approach, based on service activities performed at regular intervals, to a predictive one, that foresees maintenance intervention only when the likelihood of breakage is above an appropriate threshold, can lead to huge savings for companies in all sectors;
toxic leaks detection	timely identification of toxic compounds in air is of fundamental importance for reducing the risks associated to leakage of dangerous chemicals and it is crucial to prevent operators exposure and enhance environment protection.

In all the aforementioned cases, measures are collected over time and should be analyzed in a timely manner to extract useful information about virtually critical circumstances: traditional time series classifiers usually target the recognition rate as main goal, but this is not enough for early classification or prediction where *earliness* of decision becomes an indispensable key performance indicator.

A sequence of events that, for some reason, may end up compromising a system should be identified in the shortest possible time, as any delay could cause damages and unnecessary costs [53].

This thesis addresses the problem of on-the-fly early classification for online data streams or time series, where data are usually statistically dependent and inherently correlated over time due to causal constraints. The specific cases analyzed herein are characterized as sequential decision problem on a non-stationary data stream, which is substantially different from generic classification of time series or prediction. In the latter cases, the task can be either assigning a stream to a certain class from a complete set of values or predicting the next possible value from the preceding ones. Very consolidated statistical and autoregressive models are available, such as ARMA or ARIMA [16], that are highly effective on stationary time series, whereas the techniques for non-stationary sequences are fewer and less accurate.

Nevertheless, the problems faced in this thesis are different from the above: our aim is to be able to reliably label a temporal sequence of events, characterized by multiple heterogeneous features, using the smallest number of observations. The task is thus configured as an early decision problem, based on an incomplete set of events that requires on-the-fly evaluation and on an undefined time horizon: a critical aspect is finding the optimal trade-off between decision speed and classification accuracy, which are conflicting constraints.

Methods for early discrimination of unwanted behaviors are valuable to support fruition of such data: this document introduces and analyzes a state-of-the-art approach based on Discrete Time Markov Chain (DTMC) [23], the robust and consolidated Sequential Probability Ratio Test (SPRT) [86] and presents a novel method, inspired by the principles of quantum computing, that not only is capable of classifying data streams with outstanding accuracy but also is extremely rapid in taking the proper decision without even knowing the time horizon of the data stream. This means that my approach is completely myopic and no delay cost estimate is required to constrain early decision because it leverages the intrinsic structure of data to associate the proposed class label.

Needless to say that pre-processing of incoming data is definitely a major issue to support the actions of the proposed quantum-inspired method, but experimental results are significant and worth further insights.

The experimental validations considered so far are able to demonstrate the robustness of proposed quantum-inspired method both in binary and multinomial settings: the first one is based on the synthetic generation of multi-class probability estimates and is aimed at proving the consistency and effectiveness of the novel method, whereas the second one is a real application of binary classification to the detection of autonomous web agents, also known as bots.

On this topic, one important remark is that, to the best of our knowledge, no freely distributable datasets are available for bot detection, making it difficult to compare the presented results to other relevant studies; hence, the approach proposed in [23] has been reproduced and assessed in relation to SPRT and then SPRT has been compared with the quantum-inspired algorithm to confirm its efficacy both in terms of classification metrics and decision time.

The remainder of this document is organized as follows:

CHAPTER 2 introduces the terminology and defines the relevant meaning as used in the present document;

CHAPTER 3 presents an overview of the state-of-the-art approaches to early decision;

CHAPTER 4 describes the theoretical aspects of SPRT applied to early decision tasks, for two classes;

CHAPTER 5 introduces the theoretical background on quantum computing, which is required to understand the proposed method and its multinomial generalization;

CHAPTER 6 describes the validation process of the proposed method, based on synthetically generated data streams, and the relevant experimental results;

CHAPTER 7 defines the problem of early identification of bots from web server request logs, which is the operational context where this novel method has been first applied;

CHAPTER 8 analyzes the application of SPRT to real time bot detection from web traffic logs;

CHAPTER 9 describes the quantum-inspired algorithm application to the bot detection problem and the relevant result;

CHAPTER 10 presents concluding remarks, some cues for extending this research and outlines other areas of possible application.

2

Background Context

A data stream can define two different types of sequences that are discriminated by the presence or absence of time dependence.

An ordered list of data related by some explicit or implicit rule can be defined a *data sequence* and the order of the elements is important to represent the informational content of the data stream. The order of elements in DNA is fundamental to store the genetic code but by no means it is related to time.

Conversely, when the order of the data chunks depends on time, the stream becomes a *temporal sequence* or a *time series*. In literature [47], there is a clear distinction between the two definitions because a time series is said to be composed by points that are equally spaced in time, nevertheless, for the scope of present thesis, *data streams* will be referred to as *time series* or *sequences* interchangeably.

According to [15, 16], a *time series* is defined as a sequence of observations x_t, where each observation is associated to a specific time step t.

Time series can be categorized into *discrete time*, corresponding to a discrete set of observations recorded at a constant sampling rate, and *continous time*, where observations are continuously recorded over a given time interval. Since most processing is currently performed by means of digital calculators and a great amount of data are significant if aggregated over discrete time frames (hours, days, months, etc.), this work will be focusing on *discrete time* time series.

The sequences where each observation is described by a single scalar component are named *univariate* [UTS] whereas, when each time sample is characterized by a vector of features, the time series is defined as *multivariate* [MTS].

Analyzing a Multivariate Time Series as separate series for each feature and processing each one independently disregards the correlation among the features weakening the ability of classifiers to correctly label the stream [92].

Algorithms dealing with MTSs are specifically designed to process the whole feature set at once, which can become a major issue in case of a large number of features.

Time series analysis may have different objectives [14]:

- *prediction*: utilize the observations available at time t to forecast the values at time $t + i$; key point in this task is also to be able to specify the accuracy of forecasts and the estimate of risk associated to the relevant decision;
- *transfer function estimation*: relate an input process X_t to an output process Y_t to estimate the dynamic response of a system and eventually forecast future output values through the estimated transfer function;
- *unusual intervention events analysis*: account for the effects of the presence or absence of an event on a time series or measure the event impact in relation to its degree of presence;
- *multivariate time series analysis*: consider individual feature sequences as components of a vector time series in a joint model, thus leveraging the relationship among features at each time step to improve the accuracy of prediction;
- *discrete control systems*: implement continuous monitoring of a process with discrete time models to automatically compensate fluctuations due to noise or other irremovable causes in the input variables.

Time series *classification* refers to the process of assigning one or more class labels to a time series, using all available observations, which means that all instances have already been collected by the time of decision. Conventionally, classification implies that only one label be assigned to each sequence.

Conversely, *early* time series *classification* aims at labeling the sequence using only the shortest amount of time or the smallest number of observations, trying to assign the correct class as each observation is acquired.

It is worth noting that a *classification* task can also be used in *prediction* because a given sequence of observations may denote anomalous behaviors that end up with a system failure, thus allowing to reliably forecast abnormal operations.

An intrinsic feature of time series is that, typically, adjacent observations are mutually dependent and the nature of this dependence is of considerable practical interest: this innate characteristic is highly exploited by all analyzed algorithms, as well as by the proposed quantum-inspired method, to improve classification performance.

Moreover, another implication of early decision tasks is that each sequence has variable length as it is often impossible to estimate how many observations are required to take a reliable decision.

A time series is said to be *stationary* when the statistical properties of a process generating the temporal sequence do not change over time. This implies that its mean, standard deviation and auto-correlation remain constant over time. On the other hand, when these properties change over time, the time series is *non-stationary*.

An approach to *non-stationary* data streams is to firstly convert them into *stationary* ones, when possible, by means of such transformations as de-trending or de-seasonalization: these are easier to model and analyze, but in case of real

time classification for early decision, it is difficult to model the data source from a set of observations that changes at each new instance.

Each feature of observation x_t may be *numerical*, corresponding to a real value, in which case standardization or normalization can be applied, or may be associated to a limited set of valid symbols, therefore it is termed *categorical*, and generally requires a *one-hot* remapping to be effectively represented and used in a machine learning algorithm.

In a multivariate time series, features of both types can be found and significant pre-processing is often required, especially when the features size can pose serious limitations to the processing capabilities of the analytical platform.

Finally, with regard to problem solving strategies, an algorithm is termed *greedy* when its intent of finding a global optimum is pursued by choosing locally optimal solutions at each stage. Even if a greedy strategy does not usually yield an optimal solution, local optima are often able to approximate it at a reasonable computational cost.

3

State of the Art on Early Decision

Many natural and industrial processes generate huge amounts of data over time and, most often, time represents a way of expressing the evolution of a process and the main index on which a sequence can be ordered, analyzed and eventually classified.

Statisticians offered various methods for analyzing sequential data but most statistical models, such as ARIMA [2], assume a linear model for the data which implies that the *time series* be either stationary or convertible into stationary by applying proper transformations. However, the statistical properties of a time series most frequently vary over time, making them *non-stationary* and thus requiring new approaches [96] that are capable of building a data model from training data, such as Artificial Neural Networks (ANNs) [2].

Nevertheless, it is difficult to utilize classical machine learning techniques with sequential data because many algorithms don't take into account the autocorrelation structure of a time series and are sensitive to noise, which usually characterizes streams of data.

Despite the large number of effective time series classification approaches available in literature [47, 88], early decision is a specific aspect that requires additional focus for its unique characteristics and its importance in a whole lot of industrial applications.

Early decision is a highly valuable and complex task that aims to analyze streams of data received in real time from a data source and devise the earliest moment in time when a reliable decision can be taken, according to a given cost function. Our goal is making a reliable decision from an incomplete set of temporally related data. Decision *de facto* means classification because the act of deciding is related to selecting one option, or action class, out of all the possible ones.

Moreover, early decision can be related to *optimal stopping* theory [59] as their main goal is choosing the best time to take a given action based on sequential observations of a random variable, but it requires an estimate of misclassification or delay costs, that are usually difficult to quantify.

In the present chapter, the most interesting algorithms found in literature are reported for an extensive, if not complete, overview of the plethora of different

strategies that researchers have developed so far in a wide variety of industrial sectors and applications.

In [57], the authors present a method for classifying a time series from incomplete information and define the notion of *reliability* to describe the probability to be met when assigning an incomplete time series the same label that would be attributed to the complete data stream.

Determining the reliability of classification is the added value for this method compared to other approaches that simply assign a label to a time series as early as possible.

Considering a subsequence z representing the observations of the complete time series x and a classifier function g, the *reliability* of decision $\hat{g}$ can be estimated by means of

$$P(g(X) = \hat{g}|Z = z) = \int_{x \ s.t. \ g(x)=\hat{g}} p(x|z)dx \tag{1}$$

where x and z are modeled as random variables X and Z associated to the complete and incomplete data respectively. When the above integral is greater or equal than an *ad lib* threshold τ the classification is considered reliable otherwise it is necessary to wait for additional information and repeat the process.

Unfortunately, computing the above integral is computationally intractable, therefore an approximated conservative approach is proposed as of the following rule:

$$g(X) = \hat{g} \ for \ all \ x \in A \ for \ some \ set \ A \ s.t. \ P(X \in A|Z = z) \geq \tau \tag{2}$$

Implementing the simplified rule requires to fulfill three steps:

- estimate the conditional density $p(x|z)$,
- construct an appropriate set A,
- check whether the rule is satisfied.

This approach can be extended to multinomial classifiers and requires to estimate the mean m and covariance R of the complete data X to be applied.

The authors also refer to SPRT [86] as an alternative to the proposed method for binary classification, but point out the *greedy* connotation of this probabilistic model for sequential analysis, where the contribution of new observations does not affect the cumulative log-likelihood calculated from previous ones, as one can infer from:

$$S_k = S_{k-1} + log \ p(g(x) = 1|z_k) - log \ p(g(x) = 0|z_k) \tag{3}$$

SPRT has been extensively used in my PhD research activities on early decision as a reference Bayes-optimal approach, under the assumption that all observations are independent, and its results have been compared to those from all other considered techniques.

Considering the importance of SPRT in the context of sequential analysis and specifically for my research activities, the relevant theory is detailed in chapter 4.

Another interesting approach is proposed in [89] where the authors address the problem of early classification for some time-sensitive applications in health-care through an effective 1-Nearest Neighbor (1NN) method. The 1NN classifier has been selected because it does not require any feature selection, pre-processing, training nor configuration parameters.

Their method focuses on determining the earliest time when a reliable classification is made available, while retaining the same accuracy that could be achieved with a 1NN classifier on the complete time series. It is an instance based classifier because predictions are generated based on similarities among the input time series and the ones in the training set.

The choice of a distance measure is a critical issue but in general Euclidean distance performs better in terms of accuracy than more complex similarity measures, such as Dynamic Time Warping that can be more accurate on small data sets.

They consider a *time series* as a sequence of pairs (*timestamp, value*), ordered by timestamp in ascending order, and limit the discussion to sequences of equal length L, for simplicity.

Assuming timestamp as positive integer values, they also introduce the concept of *minimum prediction length* (MPL), representing the earliest timestamp within the training observations when the classifier begins to provide reliable class predictions for each training sequence.

At every new observation of an unclassified time series with timestamp n, the 1-Nearest Neighbor classifier is applied to the training data truncated at the same length n and the prediction is considered unreliable if the MPL is greater than n, therefore no decision is taken and a new observation is awaited.

This paper proves the effectiveness of 1NN based approaches in retaining classification accuracy while reducing the prediction time, but leaves open the key issue on how to devise the optimal trade-off between accuracy and speed of classification.

In [53], the problem of early classification is tackled with a method based on probabilistic classifiers that learn the timestamps at which accuracy begins to outperform a defined threshold. The prediction can then be issued only when timestamp is at least equal to the learned values. Moreover, in order to achieve a robust prediction, the decision is taken only if the difference between the two largest class probability estimates exceed a specific threshold that has been assigned in the training phase.

It is an Early Classification framework based on class Discriminativeness and Reliability of Predictions (ECDIRE), that analyzes how the classes are discriminated over time to learn the aforementioned decision triggers. The paper focuses on a database of time series of equal length but it specifies that ECDIRE can be applied to variable or unknown length sequences with minor changes.

The learning phase of ECDIRE is split into three tasks:

1. identify the timestamp whence a prediction can be made, for each class: this means that no attempt to predict a label is made until the minimum number of observations has been collected;

2. define a reliability value to assess predictions;

3. train a set of probabilistic classifiers to label new time series.

In the first task, a timeline is built assigning at each time step the class labels that can *safely* be predicted from that instant onward: only the classes in *safe state* are considered by the classifier, thus limiting the computational effort when processing a new time series.

At each new observation, the training of a multinomial probabilistic classifier is performed by considering the full length time series and all classes in the training set; the first instant at which accuracy meets the reliability threshold is stored for each class, even if, according to the authors, this choice could result in overfitting.

The reliability threshold is different for each class and is assigned according to the characteristics of training set, eventually referring to domain specific requirements with the ultimate goal of preserving accuracy without neglecting decision speed.

According to the authors, the timeline is a valuable tool to yield a more interpretable model of the early classification task.

In the second task, the probabilistic outputs are used to gauge the quality of predictions by setting minimum difference between the selected label and the remaining ones for each individual timestamp of the timeline.

Finally, in the third task, an ensemble of probabilistic classifiers, based on Gaussian Process models, is trained using all available training instances to subsequently classify the new sequences.

At time of prediction, the models available for the specified timestamp are considered for the *safe* classes only: any *non-safe* labeling is ignored and additional data must be awaited.

The evaluation of results takes into account two conflicting measures:

- accuracy, commonly defined as the percentage of correct classifications,
- earliness, defined by the following equation, where N is the cardinality of the test set, t_i is the decision timestamp of each sequence and L is the equal length of the full time series:

$$Earliness = \frac{1}{N} \sum_{i=1}^{N} \frac{t_i}{L} \cdot 100 \tag{4}$$

Another interesting approach is proposed in [36] for early odor identification by means of electronic nose devices. A typical application of odor classification can be found in air quality assessment where toxic leaks generate severe safety risks to the involved personnel.

This method utilizes an ensemble of serially connected classifiers, with a reject option, to analyze subsequent chunks of the sensor signal until a confident label is assigned.

If the first classifier is not able to classify the initial chunk of the signal, it defers the decision to following stage (*reject option*) that can either make a prediction or pass it on to the next stage, processing another chunk of data until a decision is taken if sufficient confidence is attained or the cost of postponing

gets too high. The cost is assigned to the value 1 if the sequence is misclassified or to a value $d > 0$ that includes both time delay and the cost of no decision, when the reject option is triggered.

Hence, in a multi-class setting where N_c is the number of classes and $c \in \{1, \ldots, N_c\}$, the *reject option* $\oslash$ for the sequence x depends on a threshold value $\tau \in [0, 1)$ such that

$$dec = \begin{cases} c & \text{if} \quad P(c|x) > \tau \\ \oslash & \text{if} \quad P(c|x) \leq \tau \end{cases} \tag{5}$$

The approach uses an *ensemble consensus* of the classifiers to accept or reject the decision therefore it is crucial that the classifiers be individually accurate and diverse to ensure a better performance of the whole system.

According to the authors of [23, 66, 88], a plethora of different techniques may be applied in early classification, but in most cases they require a vertical approach, based on very elaborated feature engineering that cannot disregard the nature of the problem itself.

One sample task, frequently referred to in early decision, is bot detection: some interesting approaches have been specifically developed on this task and will be analyzed in chapter 7.

Other approaches to early decision that can be considered of a more general purpose, reported in the previously cited documents, are based on:

- *analytical learning* feeds a machine learning algorithm with the features of each sample of the time series to estimate the likelihood that a particular sample belongs to a given class;
- *syntactical log analysis* looks for particular information in the log files to try and identify the class a specific sample belongs to.

Unlike other presented methods, these approaches do not consider the possibility of not taking a decision and late classification on the whole sequence is critical for the tasks considered so far.

Anyway, the requirement of a sequence to be entirely available before classification takes place seems to contradict the nature of the task: early decision is effective if the process that is associated to observations has not ended yet.

The vast majority of early classification methods found in literature require that the whole sequence be available before the process starts and apply to Univariate Time Series, as reported in [88, 90], where 1-Nearest-Neighbor algorithm with Euclidean or Dynamic Time Warping (DTW) distances proved to be very effective. The approach gets more complicate on multivariate sequences where alternative distance measures have to be adopted [5] to consider the correlation among features.

An alternative proposition [7, 8] suitable for multivariate time series, that leverages the correlation property, combines:

- Principal Component Analysis (PCA) based similarity measures to segment an unclassified sequence;

- a cost function to map each chunk to a real number and Dynamic Time Warping distance to train the classifier.

They first consider that Multivariate Time Series (MTS) cannot be merely examined as a collection of Univariate Time Series because the relation among features is as important as the variables themselves and carries some hidden information representing the real description of the process.

For this reason, they define a dissimilarity measure which combines the strength of DTW and PCA similarity factor to cope with the variations in the features' correlation structure.

The method, named Correlation Based Dynamic Time Warping (CBDTW), leverages the non-overlapping segmentation of a time series to take into account the alternation of latent variables and find homogeneous segments with reference to any arbitrary cost function that maps each segment into a non-negative real number. However, most of the segmentation algorithms are suitable for analyzing only one time-dependent feature but this is not sufficient when correlation among features plays a major role.

The two critical aspects of this approach are:

- the *number of principal components*, to be selected in order to guarantee a small reconstruction error;
- the *number of segments*, assigned according to the minimization of a weighted modeling error.

Another interesting approach is based on statistical analysis and relies on a penalty factor associated to decision delay, as already seen in [36]. In [19], two costs are considered to balance the quality of prediction against the speed of decision thus defining an adaptive non-myopic model. As in many other classification approaches, being non-myopic requires the whole sequence to be available at the time of decision, which is a limitation that the novel quantum-inspired methods tries to overcome without sacrificing either the quality or the rapidity of decision.

With regard to real time binary classification, one interesting algorithm, defined in [23], builds a first-order Discrete Time Markov Chain (DTMC) [12, 60] to compute the conditional probability of each class based on the likelihoods of transition patterns and initial state. In this case, a minimum number of observations must be analyzed and a decision is taken when the absolute value of the difference of the two log probabilities is greater than a given threshold. The approach will be analyzed in full details in chapter 8 as, to the best of my knowledge, is the most effective method available in literature for real time bot detection.

Another SOA model is proposed in [30], where the authors define an Early Classification Model (ECM) suitable for Multivariate Time Series, even though specifically designed for biomedical data. The novelty of this method comes from the integration of consolidated Hidden Markov Model (HMM) [62] and Support Vector Machine (SVM) [26] models that were never used before in early classification of Multivariate Time Series. HMM offers a generative model

that exploits the temporal dependencies among observations whereas SVM provides an efficient discriminative model.

This hybrid approach is applied to multivariate gene expression time series where the HMM models are in charge of estimating the log-likelihood of temporal segments and the selection of reliable estimates to classify a sequence is delegated to SVM.

The model is trained using the whole time series but it is able to take a decision on the testing data at very early timestamps with competitive accuracy, slightly lower than the best results in literature, but with only 40% of the test sequences on average.

From the same author, a completely different way of approaching early classification of Multivariate Time Series is proposed in [28], based on shapelets and still focused on biomedical informatics.

The method, called Multivariate Shapelets Detection (MSD), can achieve highly accurate classification rates analyzing up to 64% of the data sequence's length.

A multivariate shapelet is defined as a set of segments, one for each feature of the dataset, that distinctly characterize the target class and are effective for early classification. Their effectiveness is quantitatively expressed by an utility score that enables pruning the less representative shapelets.

Every new time series is classified on the best match within the shapelet dictionary according to an Euclidean distance function to be minimized across all features.

The multidimensional nature of the sequential data implies that the distance between two subsequences is expressed by a vector of real values which are related to an array of thresholds, assigned in order to maximize the information gain.

The main drawback of this approach is execution time, which grows exponentially with the size of dataset and the length of sequences, thus limiting its applicability to a reduced group of real world applications.

Another limitation of this solution lays in the definition of multivariate shapelet which is made up of multiple segments, one for each feature, with exactly the same starting and ending points. It is intuitive that distinct features might have characterizing segments that are located in different positions within the data stream, therefore being able to include the most distinctive patterns may become a competitive advantage in early classification.

This challenge is tackled in [38] with a strategy that overcomes this restriction by identifying the sub-clusters or sub-concepts that are typical of the same class label.

In order to expose the inner structure of MTS, the feature variables are analyzed independently to identify the core shapelets to be used with the classifier.

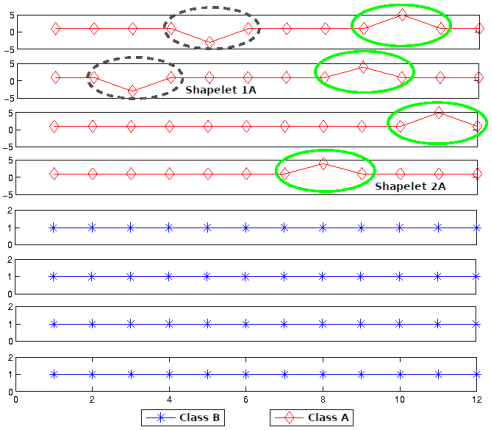
The authors observe that the length of a multivariate shapelet is a crucial aspect because, regardless of whether it is too long or too short, a wrong size may lead to non-characterizing sub-sequences.

The algorithm performs a full scan of the training sequences to dig out all the shapelet candidates for each feature variable, tuning the required precision

and recall metrics to avoid saving too many candidates. Nevertheless, only the most representative ones are used to build the early classifier.

Shapelet selection, for each class label and for each feature, is achieved by iteratively splitting the candidates into several clusters according to their Silhouette index and then merging them back into the nearest cluster, until convergence.

Figure 1: Locally distinctive shapelets



The iteration continues until core shapelets of all features are selected, including also some infrequent ones to target the problem of sub-concepts. Many shapelets can be identified for each feature but not all combinations are valid for classification.

The MTS is classified either by finding coherent and explainable rules among the features of the data stream or by a *query by committee* approach that labels each sequence component independently and assigns stream class by majority vote.

The extraction of interpretable shapelets is the major concern in [91] because some application domain expert are reluctant to trust unexplainable decisions. In this case, early classification is achieved by means of locally distinctive shapelets that, being a sub-sequence of the input stream, are effective for early classification and remain highly interpretable.

Some issues must be addressed to tackle this task:

- *identify what type of shapelets are easy to understand*: segments of the time series are the best option because they lie in the same data space of the input stream, thus allowing for better interpretability by the end users;
- *define proper criteria to devise interpretable shapelets*: sub-sequences of the time series are able to express local similarities, identified by means of a distance function;
- *find a way to build effective shapelets*: it is achieved throughout the extraction of variable length local shapelets, whence optimal ones are selected on earliness and popularity.

The aforesaid principle of locality is the right approach to answer former questions.

The example reported in Figure 1 shows that, even if shapelet 1A is shared by 2 sequences of class A, it does not cover all instances of the class, whereas shapelet 2A covers all sequences in class A and none in B therefore can be selected as representative for early classification. Nonetheless, shapelet 1A is still important for early classification of class A because it comes sooner than 2A.

A sophisticated reasoning is proposed to learn distance thresholds that are based either on Kernel Density Estimation [20] or on Chebyshev's inequality [4] and are capable of maximizing classification precision. Alternatively, it is possible to assign the desired precision and derive the relevant distance threshold.

In addition to interpretability, another important property in early classification is the ability to measure the level of confidence of class prediction that is estimating the decision uncertainty. An interesting contribution on this topic is found in [29], where the authors extend their previous works on shapelets classification by inferring the uncertainty $U(c)$ of assigning class c to a time series from a model of the relevant decision confidence $C(c)$, as of:

$$U(c) = 1 - C(c) \tag{6}$$

The confidence about a shapelet S to be able to classify a time series T is defined by two components:

- the confidence that distance between S and T is less than threshold δ;
- the precision of S in accurately classifying T.

The confidence value of a single shapelet is between 0 and 1 and, given that each shapelet has a different distance threshold, we need to consider the confidence of all shapelet matching the time series to obtain the overall $C(c)$. Hence, the confidence $C(c)$ is greater than the confidence of each individual shapelet because a sequence can be identified by more discriminating elements. This approach generally gets better classification results by waiting for a lower uncertainty shapelet to match the time series, possibly improving even more when multiple shapelets can discriminate the sequence.

To the best of my knowledge, very few classification method inspired to quantum computing are available, mainly focused on binary classification.

An original quantum inspired approach to binary classification is presented in [70], where the idea of using the quantum formalism in classical computational contexts is explored. It is a very recent publication that confirms both the growing interest on this line of research, that I've undertaken at the beginning of my PhD studies, and its promising results.

Three steps are performed to implement the quantum approach into a classical classifier, namely

- *Encoding*: a density operator is used to associate a quantum object to an equivalent *density patterns*;
- *Classification*: a quantum-inspired procedure is applied to label the *density patterns*;
- *Decoding*: a reversal operator transforms the results of classification back into the pattern domain.

Encoding of a real vector into a *density pattern* can be done in a variety of different ways, most often dependent on the some specific characteristics of the

dataset, which implies that no classifier can usually outperform its competitors in all aspects.

The selected encoding is defined through the inverse of the standard stereographic projection [69] in order to map a real d-dimensional vector X into a $(d+1)$-dimensional pure state ρ_X.

A d-dimensional vector of real numbers $X = (x_1, \ldots, x_d) \in \mathbb{R}^d$ can be mapped into $Z \in \mathbb{R}^{d+1}$ by means of the following transformation:

$$Z = \frac{1}{\sum_{i=1}^{d} x_i^2 + 1} \cdot (2x_1, \ldots, 2x_d, \sum_{i=1}^{d} x_i^2 - 1) \tag{7}$$

Hence, the relevant density pattern can be expressed as $\rho_X = Z^\dagger \cdot Z$, where the symbol † represents the complex conjugation and transposition operation.

Classification is delegated to a binary classifier designed to solve the *quantum state discrimination problem* as described in [39] by Helstrom, leveraging the property that a quantum state provides less information than multiple copies of itself.

From the density patterns of the training dataset, the *quantum centroids* for the two target classes can be defined as mixed states, which do not match the encoding of original dataset's centroids, and used to classify the test dataset.

Finally, *decoding* is possible when the encoding function is invertible, but in case of a classification process it is more meaningful to use the labels assigned to the test dataset.

All experiments were executed on binary datasets extracted from Penn Machine Learning Benchmark repository [55], that includes real-world, simulated and toy data, comparing the results against other frequently used classifiers: the new supervised algorithm proved to outperform all selected classifiers, on average.

Another quantum-inspired binary classifier, very effective on the MNIST handwritten image database, as described in [81], is based on the estimation of the density operators for each class, followed by the projective measurement of quantum states and the consistent labeling of each data element.

Despite the apparent similarities of the latter with the approach proposed in chapter 5, it does not address time series classification, nor it leverages the entanglement property for classification, confirming the innovative connotation of the classifier suggested in the present thesis.

4

The Sequential Probability Ratio Test

4.1 Theoretical background

One interesting approach to sequential analysis is based on Wald's Sequential Probability Ratio Test (SPRT) [86], which is suitable for two-class sequential decision making problems with independent identically distributed observations.

It is a sequential test of a statistical hypothesis that implements a specific rule, valid at any observation of the sequence values, in order to make one out of three possible decisions:

1. accept the *null hypothesis*
2. reject the *null hypothesis*
3. delay decision and consider an additional observation

This sequential procedure is continued until either the first or the second options are selected. The number of observations required to make a decision is not established *a-priori* but it is indeed a random variable itself, because it depends on the outcome of previous tests.

In a binary sequential test, two types of error are possible:

- error of the *first kind*: the algorithm rejects a *null hypothesis* when conversely it is true;
- error of the *second kind*: the algorithm accepts a *null hypothesis* when an alternative hypothesis is true.

The Sequential Probability Ratio Test has some optimal properties that make it interesting for early decision problems:

- it is possible to set the upper boundaries for the two types of errors;
- the minimum number of observations required to make a decision can be estimated and it is significantly smaller than with other methods;
- no probability distribution is required to execute the test.

More formally, SPRT is a sequential decision strategy that takes all observations in a data stream, one at a time, and assigns a class label $dec \in \{0, 1, \oslash\}$, where the symbol $\oslash$ means that no reliable decision can be taken and another observation must be taken into account.

Let X be a sequence of events characterized by a non-observable class label $y \in \{0, 1\}$ that has to be determined from subsequent observations $\{x_1, x_2, \ldots, x_k\}$.

Two types of errors can be defined in binary classification tasks, which can be related to:

- α: false negative rate or probability of committing an error of the first kind, that is assigning class 0 to a sequence belonging to class 1 (*null hypothesis*);
- β: false positive rate or probability of committing an error of the second kind, that is assigning class 1 to a sequence belonging to class 0.

Considering the joint conditional probability $p(x_1, x_2, \ldots, x_k | y = c)$ for $c \in \{0, 1\}$ and the two constants A and B set according to the required probabilities of error α and β, SPRT defines a decision strategy that outputs:

$$dec_k = \begin{cases} 1 & \text{if} \quad R_k \geq A & \text{\% accept the null hypothesis} \\ 0 & \text{if} \quad R_k \leq B & \text{\% reject the null hypothesis} \\ \oslash & \text{if} \quad B < R_k < A & \text{\% delay decision} \end{cases} \tag{8}$$

where R_k is the likelihood ratio

$$R_k = \frac{p(x_1, x_2, \ldots, x_k | y = 1)}{p(x_1, x_2, \ldots, x_k | y = 0)} \tag{9}$$

Optimal values of A and B are difficult to compute but can be approximated, according to two theorems from Wald's theoretical framework, whose proof is detailed in chapter 11, by considering that

- A is upper bounded by $\frac{1-\beta}{\alpha}$ and B is lower bounded by $\frac{\beta}{1-\alpha}$: in practice, Wald suggests to assign A and B to their boundary values, hence

$$A^\star = \frac{1-\beta}{\alpha}, \qquad B^\star = \frac{\beta}{1-\alpha} \tag{10}$$

- when $A^\star$ and $B^\star$ are used as an alternative to the optimal values of A and B, for the real error probabilities $\alpha^\star$ and $\beta^\star$ holds

$$\alpha^\star + \beta^\star \leq \alpha + \beta \tag{11}$$

Note that in case of independent identically distributed (IID) samples, the probability of each class c is easily obtained from

$$p(x_1, x_2, \ldots, x_k | y = c) = \prod_{i=1}^{k} p(x_i | y = c) \tag{12}$$

The rigorous application of SPRT to real problems works under the *naïve bayes*[1] hypothesis and requires that a sequence be characterized by IID samples but, even when this requirement is not verified, it is possible to apply the algorithm under relaxed constraints and achieve valuable results, as shown in chapter 8.

1 the naïve bayes hypothesis implies strong independence assumption among the features and refers to very simple Bayesian models.

5

The Quantum-inspired Entangled Multinomial Classifier

5.1 The initial idea

The current focus on Quantum technologies and their future developments on actual problems, raised my initial interest towards understanding its perspective application to my PhD studies, with particular regard to my main topic, that is time series classification.

I was fascinated and challenged by the claims on this innovative computing technology, that promises to disrupt the traditional rules of computation and bring new exceptional power to data processing.

I had the positive feeling that it would be possible to leverage in some way the basic principles of Quantum computing [64] to express dependencies and relationships among data series not detectable by any other method. But intuition is not enough to ensure valuable results are obtained, therefore I started to study the theoretical background of this new computational paradigm and devise a way to map its fundamentals principles into my research area, giving birth to a highly generalized approach that does not need to consider domain specific aspects.

The research started in the binary setting with a multivariate problem and, after probing its effectiveness on bot detection, it was subsequently extended to other multi-class applications after a deep study on synthetically generated data.

The name proposed for the classification algorithm I designed highlights its main characteristics:

- it is inspired by the principles of Quantum computing,
- it applies to multinomial univariate or multivariate problems,
- it exploits entanglement to express the correlation among the observations in the time series.

The theoretical foundations of the proposed approach are explained in the following sections where the algorithm is first introduced and explained in

the binary configuration and then generalized for its application on multi-class applications.

5.2 Theoretical background in the binary setting

Quantum computing [64] refers to the study of new computational systems that make use of quantum-mechanical properties, such as superposition and entanglement, to process data.

- *Superposition* is a fundamental principle of quantum mechanics, resulting from linearity of the solutions of the Schrödinger equation. It states that it is possible to add together any quantum states to obtain another valid state and, conversely, any quantum state can be decomposed as the sum of two or more valid states.
- *Entanglement* is a physical phenomenon that occurs when the quantum state of each of two or more particles cannot be described independently of the others.

The elementary information unit is the quantum bit or *qubit*, which is the quantum analogue of the classical bit. A qubit is a two-state quantum mechanical system that, contrary to classical bit that must be either in state 0 or 1, can be in a *superposition* of both states at the same time.

The quantum equivalent of classical 0 and 1 states is defined by the basis states of a qubit, which can be represented in the *ket* notation by $|0\rangle$ and $|1\rangle$, corresponding to the following column vectors [9, 52]:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

These two state vectors form an orthonormal basis, which means that their inner products $\langle x|y\rangle$ are:

$$\langle 0|0\rangle = \langle 1|1\rangle = 1 \quad \text{and} \quad \langle 0|1\rangle = \langle 1|0\rangle = 0$$

The above statement introduces another operator, called *bra*, that is used to write the inner product of a vector and represents the conjugate-transpose of a *ket*, defined by means of the † operator as

$$\langle x| = |x\rangle^\dagger$$

Complex-conjugation is obtained by replacing each element of a vector with its complex conjugate value and ensures that, whenever the inner product is applied to the same state, the result is a real number which can be associated with a vector length.

Hence, a pure qubit state $|\psi\rangle$ can be expressed as the superposition of all basis states

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{13}$$

where $|\alpha|^2$ and $|\beta|^2$ are the probabilities of $|0\rangle$ and $|1\rangle$ respectively, therefore

$$|\alpha|^2 + |\beta|^2 = 1 \tag{14}$$

The two coefficients of equation (13) α and β are called *probability amplitudes* and are generally complex numbers because they can hold phase information.

The tensor product, represented by means of $\otimes$ operator, is used to compute a composite state as the combination of two or more qubits [24], as in the following example:

$$|011\rangle = |0\rangle \otimes |1\rangle \otimes |1\rangle \tag{15}$$

As explained earlier in this section, superposition is a fundamental property of quantum computing, along with *entanglement*, which is a property that allows a set of qubits to express higher correlation than in classical systems. This simple consideration lays the foundation of the basic idea of this theoretical work as sequential data streams are usually characterized by an intrinsic correlation among nearby samples.

By definition, a state is considered entangled if it cannot be factorized into its more fundamental parts. In other words, two distinct elements of a system are entangled if one part cannot be described without taking the other part into consideration. A remarkable quality of quantum entanglement is that elements of a quantum system may be entangled even when they are separated by considerable space [75].

For instance, two entangled qubits in *equal superposition*, or in CatState, can be expressed by

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{16}$$

and in this case both states $|00\rangle$ and $|11\rangle$ have equal probabilities $|\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$.

The term *CatState* indicates a quantum superposition of two macroscopically distinct states and is derived from the hypothetical Schrödinger cat's experiment.
It is also possible to entangle more than two qubits in CatState as shown in the following equation:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\ldots0\rangle + |11\ldots1\rangle) \tag{17}$$

There are four postulates in quantum mechanics that specify a general framework for describing the behavior of a physical system [61]: the first two postulates are related to the superposition and measurement principles, whilst the third one defines the evolution of a closed quantum system in terms of the Schrödinger equation. The fourth postulate of quantum mechanics describes the allowable states for the composition of two or more subsystems and can be reworded as: the state space of a composite quantum system is the tensor product $\otimes$ of the state space of its components [33].

If $|\psi_1\rangle \dots |\psi_n\rangle$ describe the state of n isolated quantum systems, the state of the composite system is $|\psi\rangle = |\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle$.

The expression in equation (17) represents an entangled n-qubits state because it is not separable since it is impossible to write it as a tensor product of its original states. The advantage of qubits is that, through superposition, it is possible to encode more than one state

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |1\rangle \otimes |1\rangle = \frac{1}{\sqrt{2}}(|011\rangle + |111\rangle)$$

The proposed solution is based on these two fundamentals properties of quantum computing, as explained hereinafter.

The last aspect to consider is how to measure the probabilities of $|0\rangle$ or $|1\rangle$ from the resulting composite state, bearing in mind that the measurement process alters the state of a real quantum system[1], which turns into the pure state corresponding to the outcome of the measurement. It can be viewed as an *interface* from the quantum world to the classical one and it is the only way to extract useful information from a quantum system [33]. According to the third postulate of quantum mechanics, measurement can be performed by means of a collection of *measurement operators* acting linearly on the state space of the system, which is referred to as *projective measurement*. If a system has M valid outcomes, it is possible to define a set of $\{P_m : m \in M\}$ operators so that, if $|\psi\rangle$ is the state of the system before measurement, the probability of measuring m is given by

$$p(m) = \langle\psi| P_m^\dagger P_m |\psi\rangle \tag{18}$$

where the symbol † indicates the complex conjugation and transposition operation.

The new state of the system is then

$$|\psi_{new}\rangle = \frac{P_m |\psi\rangle}{\sqrt{\langle\psi| P_m^\dagger P_m |\psi\rangle}} \tag{19}$$

These operators satisfy the following condition:

$$\sum_{m \in M} P_m^\dagger P_m = I \tag{20}$$

and guarantee that the sum of the probabilities of all outcomes adds up to 1:

$$\sum_{m \in M} p(m) = \sum_{m \in M} \langle\psi| P_m^\dagger P_m |\psi\rangle = \langle\psi| I |\psi\rangle = 1 \tag{21}$$

For a single qubit, measurement can be done using the projectors $P_0 = |0\rangle\langle 0|$ or $P_1 = |1\rangle\langle 1|$ to obtain the probability p_0 and p_1 of the two basis states $|0\rangle$ or $|1\rangle$ respectively. Hence, the probability p_0 of a qubit being in state $|0\rangle$ can be obtained through *projective measurement* by the following equation

$$p_0 = \langle\psi| P_0 |\psi\rangle \tag{22}$$

1 Note that, in this case, the system state after measurement is not relevant as the probability $p(m)$ plays the most important role in the algorithm.

Alternatively, when the post-measurement state is not relevant as in our case, it is possible to define a *density operator* that describes the whole system [33]

$$\rho = \sum_i P_i \left|\psi_i\right\rangle \left\langle\psi_i\right| \tag{23}$$

under the two following conditions:

1. Trace Condition: $Tr(\rho) = 1$
2. Positivity Condition: ρ is a positive operator

The *Trace* is a linear operator, therefore in the case of a two state quantum system, as for the present paper, the trace condition can be written as

$$Tr(\rho) = Tr(\sum_{i=0}^{1} P_i \left|\psi_i\right\rangle \left\langle\psi_i\right|) = Tr(P_0 \left|\psi\right\rangle \left\langle\psi\right|) + Tr(P_1 \left|\psi\right\rangle \left\langle\psi\right|) \tag{24}$$

It is easy to infer that the probability p_0 of state $|0\rangle$ for the system can be expressed by

$$p_0 = Tr(P_0 \left|\psi\right\rangle \left\langle\psi\right|) \tag{25}$$

5.3 Generalization in the multinomial setting

The theoretical introduction reported in the previous section can be generalized for the classification of multinomial data streams, thus defining the structure of Quantum-inspired Entangled Multinomial Classifier (QEMC).

For a N classes problem, the reference orthonormal basis can be defined by the following column vectors:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} \quad \dots \quad |N-1\rangle = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$$

Hence, a pure *qubit* state $|\psi\rangle$ can be expressed as a superposition of the N basis states according to equation:

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle + \cdots + \alpha_{n-1} |N-1\rangle \tag{26}$$

where $|\alpha_i|^2$ represents the probability of state $|i\rangle$ and $\sum |\alpha_i|^2 = 1$

At each time step t, let $f_i(x_t), i \in [0, N-1]$ be the class conditional probabilities of the current observation x_t in the data stream.

By defining

$$\alpha_{i,t} \;=\; \sqrt{f_i(x_t)} \tag{27}$$

it is possible to represent the T subsequent observations of class i as a T-qubit entangled state $|\psi_i\rangle$ by means of:

$$|\psi_i\rangle = \alpha_i |ii\dots i\rangle = \alpha_{i,0} |i\rangle \otimes \alpha_{i,1} |i\rangle \otimes \cdots \otimes \alpha_{i,T-1} |i\rangle \tag{28}$$

For instance, the entangled state $|\psi_0\rangle$ for a hypothetical class 0 at the fifth observation can be computed through:

$$|\psi_0\rangle = \alpha_0 |00000\rangle = \alpha_{0,0} |0\rangle \otimes \alpha_{0,1} |0\rangle \otimes \alpha_{0,2} |0\rangle \otimes \alpha_{0,3} |0\rangle \otimes \alpha_{0,4} |0\rangle$$

The state $|\psi\rangle$ representing the whole data stream of length T can then be expressed as the superposition of N entangled states, each featuring some correlation among collected observations of the relevant class, according to:

$$|\psi\rangle = |\psi_0\rangle + |\psi_1\rangle + \cdots + |\psi_{N-1}\rangle \tag{29}$$

Thus, by measuring the state of the resulting quantum system $|\psi\rangle$ at every time step t, we can obtain the individual class probabilities $p_i(t), i \in [0, N-1]$ and, given a task dependent level of confidence C, make appropriate decisions as:

$$dec_t = \begin{cases} i & \text{if} \quad p_i(t) \geq C \\ \oslash & \text{if} \quad p_i(t) < C \end{cases} \tag{30}$$

If $\oslash$ is still output when the session ends, it is eventually classified as *undecided* and considered an error.

Undecided sessions appear as a separate indicator to be considered when tuning the appropriate level of confidence C. As a matter of fact, undecided sessions represent the inability of the classifier to fulfill its purpose but, even if it is clear that the correct class cannot be designated, none of the wrong ones can be elicited as most representative without committing a mistake.

Eventually, as the probabilities $p_i(t)$ measured on state $|\psi\rangle$ are normalized, for any value of C greater than 0.5, the condition expressed by equation 30 becomes necessary and sufficient for a mutually exclusive decision.

Similarly to SPRT, QEMC is also characterized as a greedy algorithm, as it tries to achieve the best classification results by analyzing local probability maxima, which are not guaranteed to be optimal overall.

PART II

Validation and Applications

The second part of this thesis describes the experimental approach used to validate the proposed quantum-inspired method and the comparison of the best-of-breed approach for bot detection found in literature with the two alternative solutions based on SPRT and QEMC respectively. Other real world applications are being tested, but results are still too rough to be reported in the present document as a significant contribution.

6

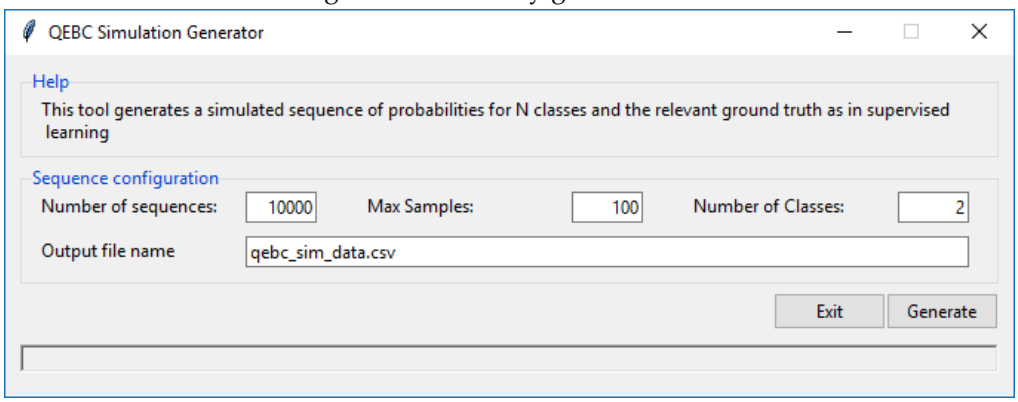
Validation of Quantum Classifier on Synthetic Data

6.1 Synthetic data generation

In order to validate the applicability of the proposed quantum inspired classifier, neglecting any dependencies on the originating task, a specific tool was developed to generate synthetic datasets of probabilities for any desired number of classes.

The tool, written in Python, offers a simple interface to define the structure of the generated dataset, as visible in Figure 2

Figure 2: Probability generator tool



The synthetic datasets simulate the results of an element-wise stream classification, therefore they contain a list of N class probabilities for a specified number of sessions having variable length up to the desired maximum number of samples.

In order to ensure a sensible bias for a specific class, every session is randomly assigned a ground truth value and, for each sample, the probability p_{true} of the *True* class is randomly taken from a continuous uniform distribution in the $[0, 1)$ interval.

Figure 3: Synthetic probability generation algorithm

Input:
    $nseq$ ▷ number of sequences
    $nmax$ ▷ maximum length of the sequences
    $nclasses$ ▷ number of classes
Output:
    $plist$ ▷ list of probabilities
procedure generate-probs($nseq, nmax, nclasses$)
    for $s = 1 \dots nseq$ **do**
        $seq\_size \leftarrow random\ integer\ in\ [2, nmax]$
        $true \leftarrow random\ integer\ in\ [1, nclasses]$
        for $j = 1 \dots seq\_size$ **do**
            $p_{true} \leftarrow random\ float\ in\ [0, 1)$
            $p_{res} \leftarrow 1 - p_{true}$
            $p\_row \leftarrow nclasses\ random\ floats\ s.t.\ \sum(p\_row) = p_{res}$
            $p\_row[true] \leftarrow p\_row[true] + p_{true}$
            $plist.append(p\_row)$ ▷ append the new row to plist
    return $plist$

The residual probability value, $p_{res} = 1 - p_{true}$, is then used in combination with a Dirichlet distribution to generate N random values that add up to p_{res}: these likelihoods are allotted to each class and p_{true} is added to the *True* class.

A numerical example, based on three classes, is reported to demonstrate the procedure used to generate each observation of a sequence:

1. let 1 be the *True* class
2. randomly assign its probability p_1; let $p_1 = 0.5614$
3. calculate p_{res} as $p_{res} = 1 - 0.5614 = 0.4386$
4. randomly define 3 probabilities that add up to 0.4386:
   $[p_0, p_1, p_2] = [0.0957, 0.0305, 0.3124]$
5. increment the element in class 1 by 0.5614:
   $[p_0, p_1, p_2] = [0.0957, 0.5919, 0.3124]$

The complete description of the probability generation algorithm is reported in Figure 3.

Even if a single event line doesn't express a clear statement on which is the *True* class, the session becomes clearly biased and this is what the algorithm is supposed to exploit in order to make a timely decision.

Table 2 displays the structure of a N classes data stream, which is saved by the tool as a Comma Separated Values (CSV) file.

Table 2: Example of generated probabilities

Session	$Class_0$	$Class_1$	...	$Class_{N-1}$	*Class Label*
0	0.49	0.01	...	0.22	0
0	0.44	0.02	...	0.19	0
1	0.03	0.12	...	0.03	1
1	0.02	0.13	...	0.02	1
1	0.02	0.21	...	0.05	1
...	...	...	...	...	...

6.2 Measuring the quantum state

In chapter 5, the measurement process for determining the qubit state has been addressed from the theoretical viewpoint, but it is also useful to add some practical considerations about the actual implementation in software.

Measurement is the only way to extract useful information from a quantum system and, in the real world, it exhibits some peculiar properties that should in principle be replicated in software simulations. These are:

1. in a real quantum system, the measurement process alters the state of the system;
2. after measurement, the system turns into the pure state associated to the outcome of measurement.

This means that, in a real system, it is impossible to estimate the likelihood of all possible basis states because, once measured, the qubit no longer contains information about the others.

Simulation software usually measures the quantum states by generating a random number and reading the associated output, which is what quantum theory would impose.

Nevertheless, in our quantum inspired case, we are not concerned about using a strictly rigorous approach to measurement and conversely we employ the *density operator* described in Equation 25 to assess the integrated probability of each individual basis state and return the top value and its associated basis state.

The normalization of the resulting quantum state before measurement takes place ensures that all probabilities add up to one and therefore the classification threshold can be constrained within zero and one.

6.3 Experimental setup

All experiments were executed on a PC Intel Core i7 3.4 GHz, with 16GB RAM, running Microsoft Windows 10 operating system with no CUDA support.

The software procedures were developed in Python language [84], at version 3, with additional support of the following standard distribution libraries: Numpy [54], Matplotlib [41], Scikit-Learn [58], Pandas [50].

Extensive testing was executed on three synthetically generated datasets, containing from two to four fairly balanced classes respectively, totaling 10.000 sessions whose individual length does not exceed 100 observations. The detailed breakdown of the sessions by class label is reported in Table 3.

Table 3: Number of sessions per class in synthetically generated datasets

Count for	class 0	class 1	class 2	class 3
2 classes	5053	4947	–	–
3 classes	3333	3335	3332	–
4 classes	2465	2526	2513	2496

The proposed algorithm has an intractable exponential space complexity due to the use of the tensor product. In order to get around this issue, a sliding window mechanism was set up to limit the number of observations considered when calculating the entangled states. This workaround was termed *peep*, for it acts as a peephole on the data stream, and it was verified by applying grid search to model tuning that *peep* values (window lengths) greater than 8, in most cases, don't bring any improvement to the overall classification scores which tend to flatten for *peep* values greater than or equal to 4. As an exception, in the binary case (2 classes), it is possible to compute the entangled states with a simpler procedure not dependent on *peep*. With more than three classes, experiments show that accuracy reaches its upper limit before exceeding the greatest bearable *peep* value, which was upper bound to the value of 10 on our target platform.

6.4 Results on synthetic data

The problem is basically setup in order to optimize two contrasting objectives:

- maximize classification accuracy
- minimize the number of observations to make a decision

A possible approach to such problem is based on multi-objective optimization, also known as Pareto optimization [56], to select the optimal threshold with regard to the selected indicators and the optimization objectives.

The possible solutions in the *decision space* are evaluated according to multiple objective functions with the ultimate goal to find a solution which is optimal in some sense.

The strategy proposed by Pareto aims at defining a set of *non-dominated solutions* that cannot be improved on one objective without degrading at least one of the others.

In the case of two objective functions, it is possible to display the solution space on a plot and visualize the set of Pareto optimal solution, also called Pareto frontier.

Grid search was applied to collect the summary indicators required to plot the relevant Pareto frontier and select the optimal solutions according to the desired objectives.

The searched parameters are:

- the confidence level C, or decision threshold DT, with values $C \in \{0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.99, 0.995, 0.998\}$
- the sliding window size with $peep \in \{4, 8\}$

For each test executed during the grid search, the following parameters and summary indicators were logged:

- DT, decision threshold on probabilities; valid values range from zero to one,
- PEEP, sliding window size,
- EXU, *exclude undecided* flag; when *True* the undecided sessions are removed from the calculation of accuracy,
- TOTSS, total number of sessions analyzed; this should be constant on all tests but it is logged to make sure all sessions have been considered,
- TOTUC, total number of undecided sessions,
- ACC, average classification accuracy, including or excluding undecided sessions according to EXU; excluding undecided sessions may take to the misleading result of 100% classification accuracy, which can be less significant because it only states that all classified sessions have been correctly recognized,
- F1, the F1 score, always excluding undecided sessions,
- PR, the precision score, always excluding undecided sessions,
- RE, the recall score, always excluding undecided sessions,
- LDS, length of the longest decision sequence; this means that at least one session required LDS steps to be classified; the average number of observations required to make a decision in the tests performed is between 3 and 5.

Table 4 reports, for a *peep* value equal to four, the parameters and their relevant metrics for those points on the Pareto front that maximize classification accuracy, minimize the number of undecided sessions or the length of the decision sequence. In order to consider the worst case, undecided sessions were included in the accuracy score.

It is evident that with low values of the decision threshold, we have contrasting results depending on the aim of Pareto optimization, whereas on more selective thresholds the performance metrics are exactly the same on both sides. On low threshold values, it is possible to zero the number of unclassified sessions, renouncing to about 5% accuracy to the advantage of decision speed,

Table 4: Classification results for 10.000 sessions with 3 classes

Aim	DT	TOTUC	%ACC	%F1	%PR	%RE	LDS
Min LDS	0.550	0	82.84	82.84	82.84	82.84	3
Min LDS	0.998	251	97.49	100.00	100.00	100.00	27
Min TOTUC	0.800	0	87.52	87.52	87.53	87.52	5
Min TOTUC	0.998	251	97.49	100.00	100.00	100.00	27

even if the greatest number of sessions is classified within the second or third observation.

At higher thresholds, the accuracy increase exceeds the 14.5% at the cost of having 251 undecided sessions, which definitely compensates the number of erroneously classified ones of the former scenarios. This situation can be seen as a limitation at first sight but, if the algorithm were analyzing a real time data feed instead of a fixed size dump file, further observations might be considered on undecided sessions and sooner or later make a reliable decision.

Another interesting viewpoint is the analysis of classifier performance, for the same settings, on an increasing number of classes, which can be easily obtained with synthetic data generator. The common settings for the reported metrics are DT = 0.995, PEEP = 4 and the results are summarized in Table 5.

The indicators reported in the header row of Table 5 have the following meaning:

- #CL, number of classes,
- OBS, number of observations,
- TOTUC, number of undecided sessions,
- ERR, classification errors,
- ACC-I, accuracy including undecided sessions,
- ACC-X, accuracy excluding undecided sessions,
- C70, number of decision steps required to classify the 70% of sessions,
- C90, number of decision steps required to classify the 90% of sessions,
- LDS, longest decision sequence,
- ADS, weighted average decision step on classified sessions

ADS is defined as the average over the total number of sequences N of all decision timestamps t_i weighted by the number of sequences classified at a given instant n_i, that is:

$$ADS = \frac{1}{N} \sum_{i=1}^{N} t_i \cdot n_i \tag{31}$$

Even if the number of undecided sessions cuts down the overall accuracy, its value stays above 97% with very few classification errors in the binary case. If we didn't consider unclassified streams, as if we could wait additional observations until a whatsoever trustful decision is made, we could ideally

reach 100% accuracy for three and four and 99.98% for two classes respectively, with only 27 observations analyzed in the single worst case.

Table 5: Classification results for 10.000 sessions with 2-3-4 classes

#CL	OBS	TOTUC	ERR	%ACC-I	%ACC-X	C70	C90	LDS	ADS
2	513601	261	2	97.37	99.98	5	8	25	4.45
3	510571	271	0	97.29	100.00	5	8	27	4.64
4	513901	111	0	98.89	100.00	3	4	11	3.08

It is noteworthy that seventy percent of classified sessions is correctly defined within the fifth observation and QEMC needs only 8 steps to classify ninety percent.

According to specific goals of the classification task, it is possible to tune the threshold to favor either accuracy or LDS, given that in all cases ADS indicator denotes high classification speed on average.

This initial experimental session pointed out an intrinsic limitation of the proposed quantum-inspired approach, allegedly due to the hardware specifications of target platform. Basically, in addition to the exploding complexity related to the sequence length, also the number of classes represents a sort of barrier that hinders the adoption of QEMC method.

On our machine, whose technical specifications are reported in section 6.3, up to 10 classes could be detected simultaneously without compromising overall system performance: alternative hierarchical approaches are possible but major changes to the proposed classification architecture are required to support two or more levels of refinement. For instance, if we were to predict possible component failures on a cyber-physical system, it would be possible to implement a first classification level capable of discriminating among the potentially affected subsystem and then pass only the involved data streams to a specialized classifier that is fine tuned for the given subsystem.

Such hierarchical approach in principle allows to cope with multinomial classification problems of any size, even on edge computers with extremely limited resources.

7

The Bot Detection Problem

7.1 Introduction

The advent of Internet and mobile technologies gave rise to a transformation in our everyday life activities, such as social live, interpersonal communication, shopping or quest for any type of information, that are moving from their mainstream connotation to virtual platforms.

Meanwhile, Web analytics and online marketing tools have been increasingly used to gain competitive advantage in this blooming market to support the creation of tailored Web-based applications capable of providing added value services and up-to-date information, collected in real time by autonomous software agents, named bot.

A *Web bot*, also known as Internet robot, Web agent, or intelligent agent, is a software program that can execute a wide variety of tasks on the net, crawling through the hyper-texts according to a predefined algorithm [27]. However, next to several harmless and useful bots, such as search engine crawlers or link checkers that support website managers in discovering broken or backlisted links, many others are inherently malignant and raise serious concerns about ethics or users' privacy as they are able to steal sensitive data, inject malware, generate Distributed Denial of Service (DDoS) attacks and many other harmful tasks.

A non negligible portion of the overall Web traffic is caused by bots and, according to [95], malware is the prevailing share among them. Bad bots tend to alter their identities by impersonating legitimate Web browsers and ignoring the file `robots.txt`, that contains website access rules for bots [31, 77, 95]. Identification of bot traffic on Web servers becomes a difficult task and the basic bot detection strategies, like matching IP addresses or user agent strings against a blacklist of known bots, are simple but sometimes ineffective approaches. A request stream may be also assessed for some atypical statistical properties, such as extremely short inter-arrival times, but these tests are often ineffective because bots tend to mimic human behavior to counteract automated identification systems.

These techniques can only be beneficial in the recognition of a limited fraction of bots, mainly the well-known ones or those whose aggressive behavior is easily characterized, like in DDoS attacks. The Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) [85] is an identification procedure and a powerful tool to contrast the aforementioned limitations with a challenge-response test that can presently be solved only by human beings, but it is quite bothering for Web users experience and sometimes limiting for people with disabilities.

Hence comes the quest for new transparent approaches that are able to tell bots and humans apart without distracting end users from their main tasks. Extensive research has been carried out on analysis, characterization and classification of robot traffic records to devise automated bot detection frameworks, even though the great majority of solutions target the *offline scenario* and the processing of historical session data. However, to the best of our knowledge, very few studies have addressed the issue of on-the-fly bot identification, while the web agent is still actively accessing the server.

Real time Web robot discovery is of crucial importance for cyber-security and Web server operations making it possible to mitigate threats before the end of bot visits and reduce the negative impact of the malicious ones. As a matter of fact, this requires new methods for early detection of Web bots in real time, based on information collected during their visits. A source of data that proved to be very informative in bot discrimination comes from HTTP request logs, which expose very representative features and relationships about ongoing sessions.

Recent studies on HTTP workload of Web servers report that robots are responsible for most of the traffic and this trend is constantly growing [23, 95]; moreover, the share of *bad bots* with malicious goals, such as impersonators, scrapers, spammers and hacking tools, is greater than 50%. Whilst good autonomous agents abide by the directives of the *robots exclusion protocol*, as of file `robots.txt`, the typical behavior of malicious bots is aimed at mingling with legitimate clients to go unnoticed through the websites, therefore only a small portion of Web bots are easily identified.

Investigations on the differences between bots and humans in Web traffic patterns, typically based on Web server access logs [21, 22, 48, 77], showed dissimilar behaviors, that pushed further research towards defining classification approaches supported by statistical analysis of navigational patterns [23, 34, 49].

In this area, most effort has been directed to the use of classification techniques, such as Bayesian classifiers [72, 79], decision trees [46, 80], support vector machines [32], association rule mining [42], or ensemble methods [71]. Some studies aimed at comparing the efficiency of various classification algorithms [13, 67, 73]. Furthermore, unsupervised classification techniques revealed a high potential for differentiating between bots and humans [74, 94].

Preliminary results on offline classification of Web sessions, reported in [65], showed that, due to some intrinsic behavioral differences, machine learning

techniques may be effective in discriminating bots from humans even in an unsupervised setting.

Offline bot detection means that, in order to take any decision, the whole navigation session must be analyzed but, even though this approach allows to gain a deeper understanding of bot traffic traits and assess its impact on server performance and security, it does not provide valuable support to detect them as they access the website and eventually apply proper enforcement policies.

7.2 Problem statement

In order to verify the proposed approaches to early detection of Web bots, traffic from a Web server hosting an e-commerce website has been used as source of data at the HTTP protocol level [11, 25], defined at the application layer of the ISO-OSI stack.

All Web clients access server resources by means of HTTP protocol, issuing a request and then waiting for a response to complete the transaction. Basically, a simple request message from a client consists of the following components:

- a request line to get a required resource, using one possible request method, such as GET, POST, or HEAD;
- headers containing some meta-information like the *user agent string* that identifies the client;
- an optional message body.

whereas the relevant response contains:

- a status code related to requested resource;
- headers;
- an optional message body.

Human users can access Web servers with browsers or mobile applications and the interaction typically implies downloading pages sequentially linked together: a set of consecutive HTTP requests is generated by a browser to access the page description file and the embedded objects like scripts, style sheets and images. Conversely, intelligent agents can traverse the site according to a specified strategy, neglecting the hyper-text structure, and possibly requesting only some types of resources.

HTTP is a stateless protocol, therefore no permanent connection is established between a server and a client: other mechanisms are available to track sessions in customers' visits, e.g. cookies [44] but they may not be standardized across Web applications nor easy to obtain from site managers.

It is then common practice [13, 23, 71–73] to define a session, whose length may vary from two to several thousands items, as a sequence of incoming HTTP requests subject to the following conditions:

1. are associated to the same IP address;

2. can be related to the same user agent string;
3. the time interval between any two subsequent requests does not exceed 30 minutes.

The problem of real-time session classification on a Web server can be stated as an instance of early classification of multivariate data streams, which means labeling each sequence in the shortest possible time. Since a session corresponds to a single visit of a given client, it is sensible to assume that, within a session, the website is accessed in a consistent style, hence the corresponding data stream might be thought of as generated by a stationary source; however, statistical dependencies in human navigational patterns, though difficult to model, are reflected in class conditional probabilities estimated by the first stage of the proposed framework, highlighting a non-stationary process.

It is then possible to formalize online Web bot detection as the task of identifying whether a sequence of HTTP requests within a session can be labeled as performed by a bot or human before the sequence ends. In principle, it is a binary classification task with sequentially sampled inputs, but the unpredictable length of sessions requires to account for a *no-decision* state whenever a session ends before the system selects a valid target label.

7.3 State of the art on bot detection

The advancements of the last few years in bot capabilities impelled the quest for disruptive detection approaches to tackle the different applications. Several methods investigate the statistical discrepancy between bots and humans behavior, detectable in some application dependent features.

For instance, [97] proposes detection of click frauds in pay-per-click advertising by identifying duplicate clicks using group Bloom filters. Another approach described in [63] leverages a set of relational, behavioral, and linguistic features to spot malware and applications subject to search rank frauds in Google Play. Artificial actors in e-dating are detected by means of specific indicators relating posted messages and the types of interaction between users. Several supervised classification techniques have been also applied to specific facets of the problem, such as Support Vector Machine (SVM) to detect spambots [37], decision trees to recognize blog bots [18], or Bayesian classifiers to deal with chat bots [31] and click frauds [87].

All cited methods are tailored for specific tasks and cannot prescind from application domain knowledge to select custom features, whereas more independent approaches based on HTTP traffic collected at Web servers resulted good at discovering bots [22, 48]. Analyses of data records from Web server access logs [13, 21, 77, 80], display evident discrepancies in navigational patterns of bots and humans but bot requests have some representative traits that could be used to improve classification rates, such as requiring less images, different order in resource access [23, 45, 78], lower data volume in HTTP responses and an increased rate of unassigned referrers, HEAD or erroneous requests.

Most bot detection algorithms that implement traffic pattern analysis require that the whole session be available to perform an *offline* classification, supported by probabilistic models built upon some statistical properties of HTTP requests.

Conversely, when machine learning is used, the applied techniques may differ by selected features, algorithms, or session extraction paradigm or even experimental evaluation.

Supervised session classifiers have been implemented with decision trees [6, 13, 32, 46, 73, 80], neural networks [13, 65], logistic regression [13], support vector machines [32, 42, 65, 67, 73], Bayesian classifiers [42, 67, 73, 79], k-Nearest Neighbor (kNN) [67, 73], and ensemble methods [71]. Unsupervised classification approaches have used several algorithms such as k-means and Graded Possibilistic c-Means (GPCM) [65], Particle Swarm Optimization (PSO) clustering [3], Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [93], Self-Organizing Maps (SOM) [35, 74], Modified Adaptive Resonance Theory 2 (ART2) [74], and Markov clustering (MCL) algorithm [94].

The prominent results achieved with the aforementioned literature approaches reasserted the inherent variations on bot and human generated traffic, even if some bots tend to imitate human behavior to avoid being discovered, making it more difficult to recognize covert Web crawlers than those which accomplish some explicit attack [1, 40, 43].

Concerning real-time detection, very few studies investigated the problem of on-the-fly bot detection and generally a constraint on the minimum number of requests to be observed before taking a decision is set, limiting their efficacy on very short sessions.

In this context, the most relevant study is [23], which proposes an approach based on a first-order Discrete Time Markov Chain (DTMC) [12, 60] model that analyzes transition patterns of resource requests to label each session in real time, achieving significant classification accuracy. For this reason, it has been selected for a comparative study versus the approaches developed during my PhD.

The algorithm is based on a first-order Discrete Time Markov Chain (DTMC) [12, 60], used to compute the conditional class probability of each request pattern. Every state of the DTMC represents a resource aggregation and, for each possible decision, a transition matrix encodes the probabilities p_{ij} that a resource of type i is followed by another of type j. Considering s_k as the probability of a session to begin with a resource of type k, the probability of a session being human or robot generated is then computed, at every new request, by multiplying s_k by the p_{ij} of all the following resource transitions. To prevent possible issues due to small probability values, the log-probabilities are used and the product is replaced by a sum.

The session could therefore be classified according to the highest logarithmic probability between the classes but, in order to guarantee a reliable decision, two new parameters must be taken into account to control detector's selectivity; these are:

- k, the minimum number of request that should be analyzed before making a decision
- Δ, which expresses the minimum difference that must be present between the class log-probabilities

A decision is taken only after k request have been received at the web server and when the absolute value of the probabilities difference is greater than Δ.

Other studies labeled the sessions leveraging decision trees but considering the user clicks on web pages instead of the stream of HTTP requests, like in [80], where a reasonably high accuracy on early decision was determined according to a minimum number of page requests to be considered to identify Web bots. Also in [6], a fixed minimum number of page requests must be observed before a decision can be taken, hence their performance on a real Web server dataset not available. My approach aims at overcoming the limitation on the minimum number of observations by defining early decision algorithms that are capable of assigning a reliable classification based on a partial incremental view of user agents' behavior, as expressed by the log records.

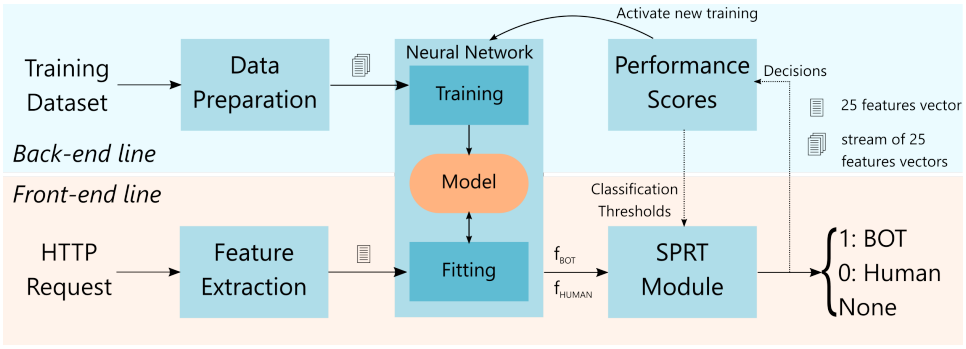
This goal is reflected also in the classification scores which are evaluated in terms of the number of requests sufficient to classify a session.

7.4 Methodological framework

7.4.1 *System architecture*

The general framework of the proposed approach, shared between both classification methods, is illustrated in Figure 4 and it is logically divided into two main processing lines.

Figure 4: A graphical sketch of the implemented framework.



The *back-end line* gathers all activities that are performed periodically, activated by specific performance considerations, such as pre-processing of historical HTTP log records, training of Artificial Neural Network model, retrieving the classification scores to detect downgrades in performance and eventually trigger a re-training phase on the ANN.

The *front-end line* collects activities performed on-the-fly for each incoming request or, more specifically, feature extraction and two-stage classification. At each new request within a session, the ANN estimates its class conditional probability which is used to tentatively make a decision about a target class. If the confidence on possible decisions is too low a new request is awaited to integrate additional information but whenever a session ends before deciding, it is labeled as *undecided* and possibly processed according to some application-specific policy: for instance, a CAPTCHA might be displayed to manually verify the user agent, or off-line analysis is applied.

7.4.2 *Request features used in classification*

As shown in the *front-end* processing line of Figure 4, the HTTP request headers supply ready-to-use descriptive features, listed in Table 6, which can have any out of three data types, each requiring different pre-processing actions:

- numerical features (N) are standardized by subtracting the mean and scaling to unit variance;
- categorical features (C) are *one-hot* encoded, i.e., represented as a bit vector of all zeros except one
- boolean features (B) are represented by 0 for `false` and 1 for `true`.

Although data for real-time processing are obtained directly from requests, historical data for classifier training may also be collected from log databases maintained at the server (Figure 4, *back-end* line).

After feature extraction and pre-processing, each request at the server is represented as a 25-feature vector, which is then submitted to the two-stage classification process.

7.4.3 *The idea behind two-stage classification*

The task under consideration is a binary classification problem whose goal is to state whether a sequence of observed Web server requests is relatable to a human or a bot agent.

Even if the HTTP protocol has no notion of session and each request is handled independently, their order is by all means dependent on the visiting agent [80].

At time point t, let $x_1, \ldots, x_t$ be a sequence of observations from the same user agent and $c \in [0, 1]$ the possible target labels, then the probability $p_1(t)$ that the sequence belongs to class 1 is

$$p_1(t) = \frac{\Pr(c = 1)}{\Pr(x_1, \ldots, x_t)} \cdot \Pr(x_1 | c = 1) \cdot \prod_{i=2}^{t} Pr(x_i | x_1, \ldots, x_{i-1}, c = 1).$$

Table 6: Original Request Features before Pre-Processing

Name	Type	Description
inter_arrival_time	N int	Time interval between timestamps of the current request and the preceding one (in seconds)
method	C string	HTTP method specifying an action to be performed on a given resource (e.g., GET, HEAD)
response_status	C int	HTTP response status code (e.g., 200, 403, 404)
response_size	N double	Volume of data in the HTTP response (in kilobytes)
is_referrer_empty	B bool	Whether the HTTP referrer is known (`false`) or not (`true`)
is_page	B bool	Whether the requested resource is a page description file (`true`) or an embedded object file (`false`)
is_graphic	B bool	Whether the requested resource is a graphic file (`true`) or not (`false`)
is_script	B bool	Whether the requested resource is a script/-program file (`true`) or not (`false`)
is_style	B bool	Whether the requested resource is a style sheet file (`true`) or not (`false`)
is_datafile	B bool	Whether the requested resource is a specific data file (e.g., a zipped file) (`true`) or not (`false`)

This model however has intractable complexity, thus it is commonly simplified by considering, according to Markov assumption, a fixed number of past observations, usually one:

$$p_1(t) \propto \Pr(x_1|c=1)\prod_{i=2}^{t}\Pr(x_i|x_{i-1},c=1).$$

The above model, presented in section 8.2, is the basis for the reference method [23] selected for comparative evaluation despite the authors followed a *naive* variation that neglects the conditional dependencies over subsequent observations, leading to:

$$p_1(t) = \prod_{i=1}^{t}\Pr(c=1|x_i).$$

At first stage, each individual HTTP request is assessed to estimate a likelihood of being bot or human generated, then at the second stage a sequential

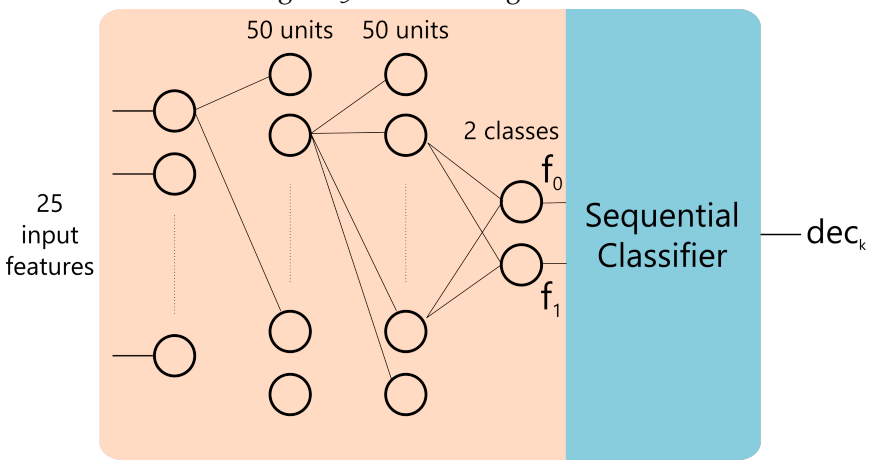
classification approach is used as a *probability integrator* for multiple observations such that we take the final decision as soon as the degree of confidence is satisfactory.

The *naive* assumption corresponds to the hypothesis that, given an appropriate description, looking at the mix of request types is sufficient to discriminate between bots and humans. Experiments confirm that this approach achieves very good results, outperforming the reference while featuring less model parameters to be fitted.

7.4.4 *The two-stage classification model*

Figure 5 depicts the schematic representation of classifier's main building blocks, made up of an Artificial Neural Network juxtaposed to the sequential classification model, which can be either the Sequential Probability Ratio Test or Quantum-inspired Entangled Multinomial Classifier. The advantage of this solution is that both blocks could be replaced by any equivalent one, providing that the interface between them, defined as a sequence of estimated class conditional probabilities, remains the unchanged.

Figure 5: The two-stage model



While the alternative *Stage 2* models are analyzed in depth in the following chapters, a brief overview of the first stage is sufficient to understand how the input features are transformed to obtain an *a-posteriori* class conditional probabilities for each individual observation.

The ANN at the first stage outputs an estimate of likelihood $f_{bot}(x_t)$ that any given observation x_t belongs to class *bot*.

The corresponding $f_{hum}(x_t) = 1 - f_{bot}(x_t)$ is the likelihood of x_t being from human. This is done without considering any context information, regardless of any previously received request, according to the *naive* assumption previously described.

Experimental considerations on classification metrics led to selecting a multilayer perceptron, whose architecture has been empirically optimized both in terms of number of layers and of hidden units: it has a 25-unit input layer, cor-

responding to the input features size, followed by two 50-unit hidden layers with ReLU activation function and cross-entropy as cost function for training. To output the class-conditional likelihood, the final layer uses *softmax* units.

The second stage of the classifier's architecture is responsible for managing the sequential nature of sessions in order to try and label the time series at every new request. The modular structure of the system supports experimentation of multiple algorithms based on the same estimated likelihoods that are treated as sequences of *a-posteriori* probabilities by the two approaches selected.

7.4.5 *Dataset description*

Access logs from an online bookshop[1] provided the HTTP requests database for our experiments. The online store offers a plethora of different physical and digital items, such as books or computer games but also audiobooks or multimedia content.

With regard to the technical details, the website is based on the osCommerce platform, hosted on a Linux server with the LAMP (Linux, Apache, MySQL, PHP) open source stack. The site access data are recorded according to *NCSA Combined* log format and cover the period from April 1st to 30th, 2014. The whole request dataset contains 1 397 838 HTTP request entries, totaling 13 395 sessions reconstructed according to the procedure described in section 7.2. Sessions with only one request were discarded due to their negligible informational content.

Additional details about dataset preparation and class breakdown will be provided in subsection 8.3.1.

1 The website identity cannot be revealed, due to a non-disclosure agreement.

8

Bot Detection with SPRT

8.1 Sequential classification

As introduced in the previous chapter, the second stage of the classifier acts as a probability integrator to devise a sufficient degree of confidence about the decision to be taken, whenever possible.

In the present chapter, we will discuss the benefits of applying Wald's Sequential Probability Ratio Test (SPRT) [86] to estimate the class posterior probability while new requests are received by the system.

The original test is based on the assumption of statistical independence among the observations therefore it is considered suitable for the case at hand, which can be simplified under relaxed constraints.

As explained deeply in chapter 4, the application of SPRT to bot detection is implemented considering the k-th observation x_k in input at the first stage, to gauge the posterior probability $f_1(x_k)$ of class 1 (bot) and consequently compute $f_0(x_k) = 1 - f_1(x_k)$ for class 0 (human).

The ratio of these probabilities at step k is:

$$R_k = \frac{p_1(k)}{p_0(k)} = \frac{\prod_{i=1}^{k} f_1(x_i)}{\prod_{i=1}^{k} f_0(x_i)} = \prod_{i=1}^{k} \frac{f_1(x_i)}{f_0(x_i)} \tag{32}$$

or, to reduce sensitivity and improve numerical precision in case of strongly imbalanced values, it can be transformed in terms of cumulative log-likelihoods as:

$$L_k = \log p_1(k) - \log p_0(k) = \sum_{i=1}^{k} \left(\log f_1(x_i) - \log f_0(x_i)\right). \tag{33}$$

Given two pre-defined threshold values T_0 and T_1, $T_0 < T_1$, defined according to probabilities of false positives α and false negatives β by Equation 10, the decision output by sequential classification stage at step k is:

$$dec_k = \begin{cases} 1 & \text{if} \quad L \geq T_1 \\ 0 & \text{if} \quad L \leq T_0 \\ \oslash & \text{otherwise.} \end{cases} \tag{34}$$

where the symbol $\oslash$ indicates an *undecided* state that implies considering additional observations, if available.

Under ideal assumptions, the SPRT approach is guaranteed to converge to a decision with the minimum number of observations for a given pair (α, β) but in this case the efficacy is weakened by two issues:

- the number of request in a stream cannot be controlled and may not suffice to make a decision at the desired confidence level, thus causing a three-state output due to the reject option for *undecided* sessions [17];
- the probability estimates retrieved with the *black-box* approach at first stage, introduce a non-quantifiable error in the values of f_1 and consequently of f_0, along with possible additional errors in the evaluation of L_k due to the *naive* assumption

The first issue impacts validation of experimental results, as of section 8.4, whereas the second one determines an approximation error in Equation 10 that cannot be evaluated in advance therefore a better choice for values of T_1 and T_0 can be attained from the training data (see subsubsection 8.1.0.1).

8.1.0.1 *Optimal thresholds selection*

The selection of threshold values T_1 and T_0 is crucial to balance classification confidence with earliness of decision: it is a multi-objective classification problem, that can be approached considering the *Pareto optimality* concept [51, 56].

Rather than arbitrarily composing multiple scores into a single objective, a search space across possible solutions is explored, highlighting the non-dominated ones that represent the so-called *Pareto Frontier*. A non-dominated solution cannot improve one objective without degrading the quality of the second objective. The *Pareto Frontier* is a powerful tool to select the method-specific parameter values *a-posteriori* in order to achieve the desired balance between objectives [68].

Section 8.3.3 describes this process and includes plots for the two methods.

8.1.1 *The algorithm*

The algorithm in Figure 6 describes the decision process that takes the t-th HTTP request of a session as a multivariate input $\mathbf{x}_t$ to obtain the class likelihoods $f_1(x_t)$ and $f_0(x_t)$ from procedure NNET. The likelihoods for observations from 1 to k are then used to compute L according to Equation 33 which is compared to the selected threshold values T_1 and T_0 to decide one out of three possible options. If no decision can be taken at step t, further observations are considered until a reliable decision is made or the session ends, in which case it is marked as *undecided*.

Figure 6: SPRT bot detection algorithm

Input:
    $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_t\}$ ▷ session (seq. of t requests)
    T_1 and T_0 ▷ the decision thresholds
Output:
    dec ▷ decision on session
    k ▷ number of steps taken
procedure classify-session(S)
    $L \leftarrow 0$
    for $k = 1 \ldots t$ **do**
        $dec, L \leftarrow$ classify-request($\mathbf{x}_k, L$)
        if $dec \neq$ None **then**
            return dec, k
    return $\oslash, t$ ▷ undecided session
procedure classify-request($\mathbf{x}, L$)
    $f_0(\mathbf{x}), f_1(\mathbf{x}) \leftarrow$ nnet($\mathbf{x}$)
    Compute L using (33)
    if $L \geq T_1$ **then**
        $dec \leftarrow 1$ ▷ decision is bot
    else if $L \leq T_0$ **then**
        $dec \leftarrow 0$ ▷ decision is human
    else
        $dec \leftarrow$ None ▷ no decision
    return dec, L
procedure nnet($\mathbf{x}$)
    Estimate $f_1(\mathbf{x})$ with neural network
    Assign $f_0(\mathbf{x}) = 1 - f_1(\mathbf{x})$
    return $f_0(\mathbf{x}), f_1(\mathbf{x})$

8.2 A reference bot detection method

In order to fairly evaluate the proposed early classification approaches, the only method that, to the best of my knowledge, can be compared to my proposals is presented in [23], a recently published paper that uses HTTP request features to try and achieve the earliest possible classification with reasonable accuracy.

The reference approach has been reconstructed and applied exactly to the same datasets used in our experiments.

The authors proposed a technique based on a Discrete Time Markov Chain (DTMC) to model the resource request patterns for the two classes. Each server asset is uniquely identified by its Universal Resource Identifier (URI) and it is categorized into a summary resource type, depending on its file extension. For the e-commerce site under examination, not all types analyzed in the original paper were present, nonetheless a consistent taxonomy devoid of unused directory content is:

- *web* for web page and script files (e.g., html, htm, php, cgi, asp, jsp, js),
- *text* for text-formatted files (e.g., txt, xml, sty, tex, c, cpp, java, css),
- *doc* for rich-text documents (e.g., doc, xls, ppt, pdf),
- *img* for images (e.g., bmp, jpg, png, tiff, raw, ico),
- *av* for multimedia files (e.g., avi, mp3, mpg, au),
- *prog* for program files (e.g., exe, dat, bat, dll, msi, jar),
- *compressed* for compressed files (e.g., zip, gz, 7z, rar),
- *malformed* for malformed requests or unknown file extensions.

Basically, a session is represented as a sequence of resource types associated to each HTTP request, which defines a *resource request pattern*, modeled through a first-order Discrete Time Markov Chain in the state space of resource types contained in the training dataset.

The model is described by the pair $(\mathbf{s}, \mathbf{P})$, where $\mathbf{s}$ is a vector whose elements s_i are the probabilities that a session starts with a resource of type i and $\mathbf{P}$ is a matrix whose elements p_{ij} are the transition probabilities that a resource of type j follows a one of type j.

Let $\boldsymbol{S} = (x_1, \ldots, x_t)$ be the resource request pattern of a session with t requests observed on the server. The probability that a DTMC will generate $\boldsymbol{S}$ at step $k \leq t$ can be, defined by:

$$\boldsymbol{Pr}(\boldsymbol{S}|\mathbf{s}, \mathbf{P}) = \mathbf{s}_{x_1} + \sum_{i=2}^{k} p_{x_{i-1}, x_i} \tag{35}$$

Being a binary classification problem, at training phase, two separate DTMC models are built, $\mathbb{R} = (\mathbf{s}_r, \mathbf{P}_r)$ and $\mathbb{H} = (\mathbf{s}_h, \mathbf{P}_h)$, for bots and humans respectively, so that we can estimate the corresponding probabilities $\Pr(\boldsymbol{S}|\mathbb{R})$ and $\Pr(\boldsymbol{S}|\mathbb{H})$.

The algorithm reported in Figure 7 requires only two parameters to be set:

- $k_{\min}$, which is the minimum number of requests that must be considered before any decision to contrast fluctuations in probabilities on the very first requests;
- Δ, which is the threshold to be exceeded to make a decision for a session.

Figure 7: DTMC bot detection algorithm

```
1:  Input:
2:      S = {x_1, x_2, ... x_t}                    ▷ session (seq. of t requests)
3:      k_min and Δ                                ▷ the decision parameters
4:  Output:
5:      dec                                        ▷ decision on session
6:      k                                          ▷ number of steps taken
7:  procedure classify-session(S)
8:      for k = 1 ... t do
9:          dec ← classify-request(x_k)
10:         if dec ≠ None then
11:             return dec, k
12:     return ⊘, t                                ▷ undecided session
13: procedure classify-request(x)
14:     update Pr(S|ℝ) and Pr(S|ℍ) for new x
15:     if k ≥ k_min then
16:         compute D = log Pr(S|ℝ) / Pr(S|ℍ)
17:         if |D| > Δ then
18:             if D ≥ 0 then
19:                 dec ← 1                        ▷ sign is positive, decision is bot
20:             else
21:                 dec ← 0                        ▷ sign is negative, decision is human
22:             return dec
23:     return None
```

Similarly to the algorithm in Figure 6, whenever a session ends before a classification is made, it is labeled as *undecided*, but the main difference from my approach is that in this case *undecided* sessions are passed to an *offline* classifier that takes all observations into account. Even though this step is sensible from the performance evaluation perspective, it is not viable when applied to a real-time environment where timely decision is crucial, hence those sessions are left *undecided* for a fair comparison of the two methods.

8.3 Experimental evaluation

8.3.1 *Data preparation*

The first and most important step for the preparation of our dataset is the allocation of a ground truth label, as bot or human, to every observation of each session. This task is by no means easy and takes a lot of manual effort: it has been accomplished by inquiring two online databases containing user agent strings and IP addresses, publicly recognized by experts as representing either bots or legitimate Web browsers, encompassing also heuristics concerning session features indicative of a bot [78].

A primary source was the Udger online database [82], containing 2832 and 843 known user agent strings of bots and browsers respectively, as well as 996 657 known bot IP addresses. A supplementary source was User-agents online database [83] with 2459 known user agent strings.

According to information retrieved, a few labeling rules were stated:

1. A session was labeled as bot if at least one of the following conditions was met:
   - the user agent was classified in the Udger database as "crawler", "e-mail client", "library", "validator", "multimedia player", or "offline browser";
   - the user agent was classified in the User-agents database as a robot;
   - the user agent contained a keyword suggesting a bot, such as "spider", "crawler", "robot", "worm", "search", "track", "harvest", "hack", "trap", "archive", "scrap", etc.;
   - the IP address was classified in the Udger database as "crawler", "fake crawler", "known attack source – http", "known attack source – mail", or "known attack source – ssh";
   - at least one of the following indicators was true: zero image to page ratio, 100% of page requests with empty referrers, 100% of response status codes of type 4xx, or 100% of requests with HEAD method;
   - the file robots.txt was requested.
2. A session was labeled as human if the user agent was classified in the Udger database as "browser" or "mobile browser".

Any other session that didn't match the listed conditions remained unlabeled and were subsequently excluded from the dataset. The final breakdown of the labeled dataset was 6190 bot sessions and 7200 human ones which means the classes were roughly balanced for our analysis.

The proportion between classes was preserved when training the ANN estimator with 10-fold cross-validation technique over data splits containing 619 bot and 720 human sessions respectively.

8.3.2 *Performance evaluation*

The experiment conducted for both SPRT and DTMC approaches on the dataset described in subsection 8.3.1, using the same training and test splits, were compared on the same scores with regard to the following scenarios:

> **Scenario 1:** Performance scores are determined only on account of the sessions classified as bot or human: no *undecided* sessions are considered.
>
> **Scenario 2:** Performance scores are determined taking into account *undecided* sessions. Since the goal is bot detection, *undecided* sessions can be considered a failure of the classifier in detecting bots, hence they are always labeled as human-generated.

The overall classification results are reported by means of a confusion matrix, considering bot-generated sessions as *positive* and human-generated ones as *negative*; as explained, undecided sessions are ascribed to humans and therefore included into *negatives*, thus increasing FN with undecided bot sessions and TN by undecided human ones. However, the number of *undecided* sessions is still reported as an additional measure of classifiers' performance.

The following metrics were used to assess classifiers performance, averaged over the 10 cross-validation folds:

- $Recall = \frac{TP}{TP+FN}$, fraction of positive sessions that are correctly classified;
- $Precision = \frac{TP}{TP+FP}$, fraction of positive decisions that are correct;
- $F1$, harmonic mean of recall and precision, overall quality of the classifier;
- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$, fraction of correct classifications;
- k_{90}, the 90^{th} *percentile of* k, maximum number of steps required to classify 90% of non-undecided sessions; indicates the ability to take up early decisions;
- $P_c(k)$, percentage of sessions classified at step k;
- $P_u(k)$, percentage of undecided sessions that ended at step k;
- $CP_c(k) = \sum_{i=1}^{k} P_c(k)$, cumulative percentage of sessions classified in less than k steps;
- $CP_u(k) = \sum_{i=1}^{k} P_u(k)$, cumulative percentage of undecided sessions that ended in less than k steps;
- P_c, percentage of classified sessions.

8.3.3 *Tuning the parameter values*

As already explained, two threshold values, T_0 and T_1, must be tuned for the SPRT based method to balance the trade-off between confidence in classification and earliness of decision. This impacts the metrics listed in the previous section with particular regard to maximization of $F1$, preferred to Accuracy due to the slight imbalance of the dataset, and concurrent minimization of k_{90}. Similar considerations apply to the selection of proper values for $k_{\min}$ and Δ in the reference DTMC approach.

Figure 8: Pareto frontier for SPRT.

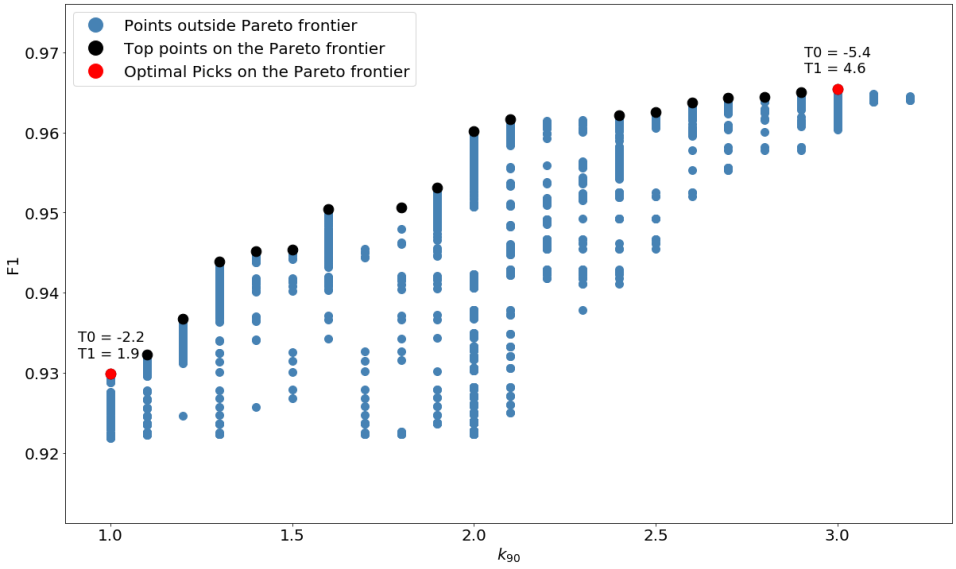
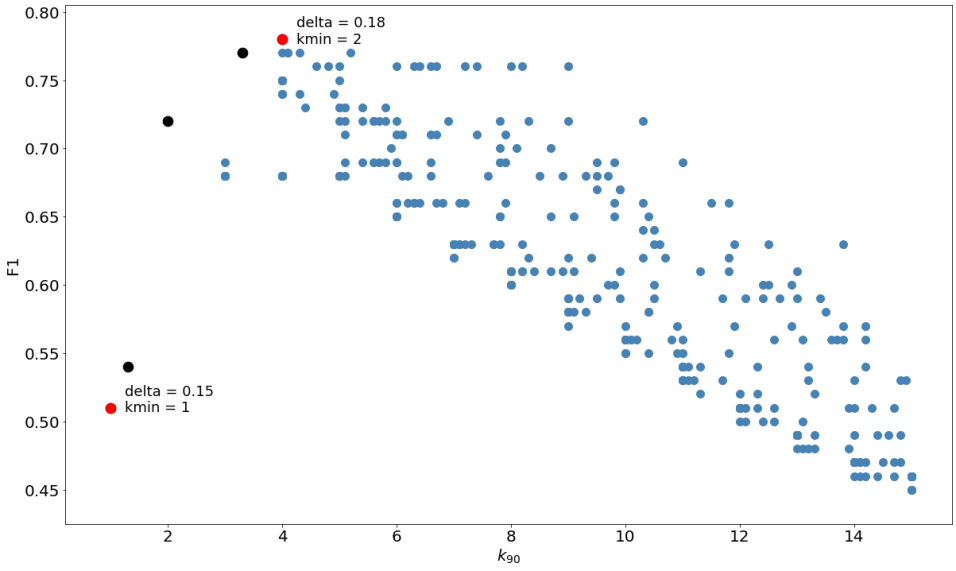


Figure 9: Pareto frontier for DTMC.



A grid search was performed for SPRT, with T_0 ranging from -5.5 to -0.2 and T_1 from 0.1 to 5.4, recording the solutions in order to build a bi-dimensional search space over the two contrasting objectives (maximization of $F1$ and minimization of k_{90}) that form our Pareto frontier.

Analogous analysis was done on DTMC where the search space was set depending on varying values of Δ, from 0.01 to 1.9, and $k_{\min}$, from 1 to 21. Figure 8 and Figure 9 report the resulting Pareto frontiers for both methods. Please note that only values of Δ from 0.01 to 0.29 are displayed for DTMC because greater values led to unacceptably low values of $F1$.

Those tuning experiments were carried out for scenario 2 which can be considered the most detrimental because it also includes undecided sessions in the computation of performance metrics.

As displayed in the SPRT plot, our solution yielded extremely good results: for all combinations of the threshold values, $F1$ exceeds 0.92, which indicates very high rates of both recall and precision. Moreover, k_{90} is stably below 3.2, which means that a decision was made well before the 4^{th} request in 90 percent of classified sessions. The extreme points on Pareto frontier, which indeed contains 18 non dominated solutions, are those where either the speed of decision or $F1$ are the highest. For $T_0 = -2.2$ and $T_1 = 1.9$, the best earliness was achieved with k_{90} equal to 1 and the corresponding $F1$ equal to 0.93, whereas maximum accuracy equal to 0.96 was gained for $T_0 = -5.4$ and $T_1 = 4.6$ with a corresponding k_{90} equal to 3.

Conversely, the plot for DTMC show much higher sensitivity to the variations of its parameter values, with performance scores much worse than for SPRT. Even if its Pareto frontier covers a more extensive set of values over the two parameters, only the best solutions are plotted, for $F1$ greater than 0.4 and k_{90} lower than 17. Some points are overlapping: for $k_{\min} = 1$ and Δ ranging from 0.01 to 0.15 there are 15 coincident solutions with minimal k_{90} equal to 1 and a very poor $F1$ equal to 0.51. The maximum $F1$, equal to 0.78, was obtained with $k_{\min} = 2$ and Δ ranging from 0.18 to 0.2 with the relevant k_{90} equal to 4.

8.4 Results and discussion

In this section, Pareto optimization [56] is exploited to determine the optimal values for method-specific parameters and subsequent analysis of classification results is accomplished with an eye to computational overhead and implementation issues.

Out of the Pareto-optimal solutions, direct comparative analysis was made selecting the options with maximum value of $F1$, including additional criteria on highest accuracy and smallest number of *undecided* sessions to settle equivalent points. According to this formulation, the parameter values for experimental phase have been set as $T_0 = -5.4$, $T_1 = 4.6$ for SPRT and $k_{\min} = 2$, $\Delta = 0.18$ for DTMC.

Table 7 reports a summary of the overall results, in order to support our analysis on decision earliness of the two methods, with regard to the rates

Figure 10: $CP_c(k)$ – Cumulative percentage of classified sessions

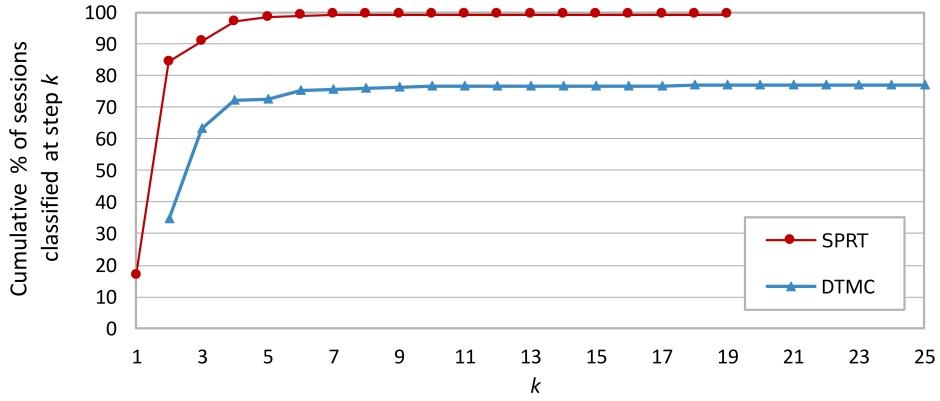


Figure 11: $P_c(k)$ for SPRT

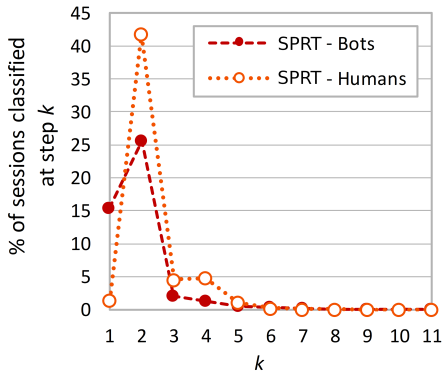


Figure 12: $P_c(k)$ for DTMC

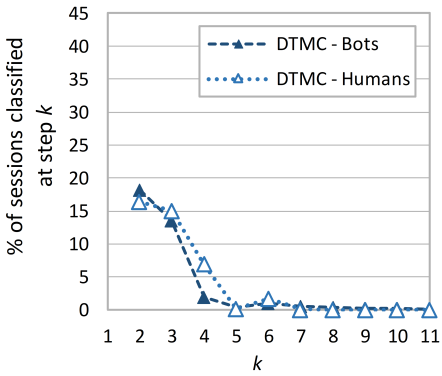


Table 7: Classification results (10-fold cross-validation)

Metric (avg.)	**SPRT**	**DTMC**
#TP	577.3	429.2
#TN	710.5	494.7
#FP	9.5	50.4
#FN	32.5	58.5
#undecided – bots	9.2	131.3
#undecided – humans	0	174.9
k_{90}	3.0	4.0
P_c	99.31	77.13

of classified and undecided sessions. The SPRT algorithm was able to make a decision on nearly all active sessions, leaving undecided only 0.69% of the them, that were incidentally bot generated. The rate of undecided sessions is much higher for DTMC where 22.87% of test dataset was not classified before sequence end, further detailed in 21.2% bots and 24.3% humans.

Figure 10 shows cumulative percentage of sessions classified with reference to the number of requests processed before making a decision. As reported,

Figure 13: $CP_u(k)$ – Cumulative percentage of undecided sessions

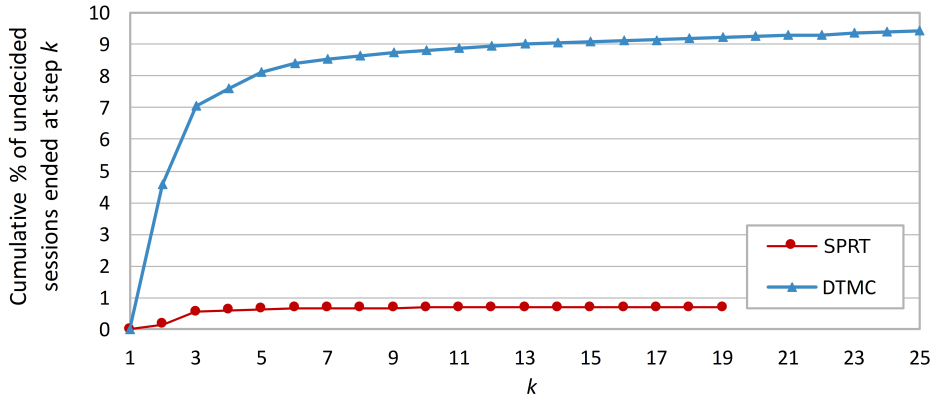


Figure 14: $P_u(k)$ for SPRT

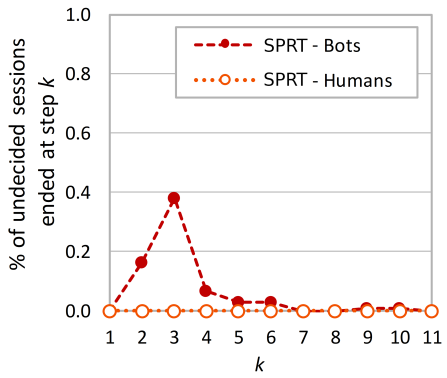
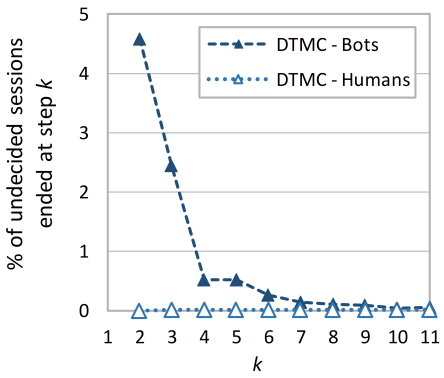


Figure 15: $P_u(k)$ for DTMC



SPRT was able to classify 99% of sessions within six requests, with 16.7% being classified at the first step and another 67.5% at the second step.

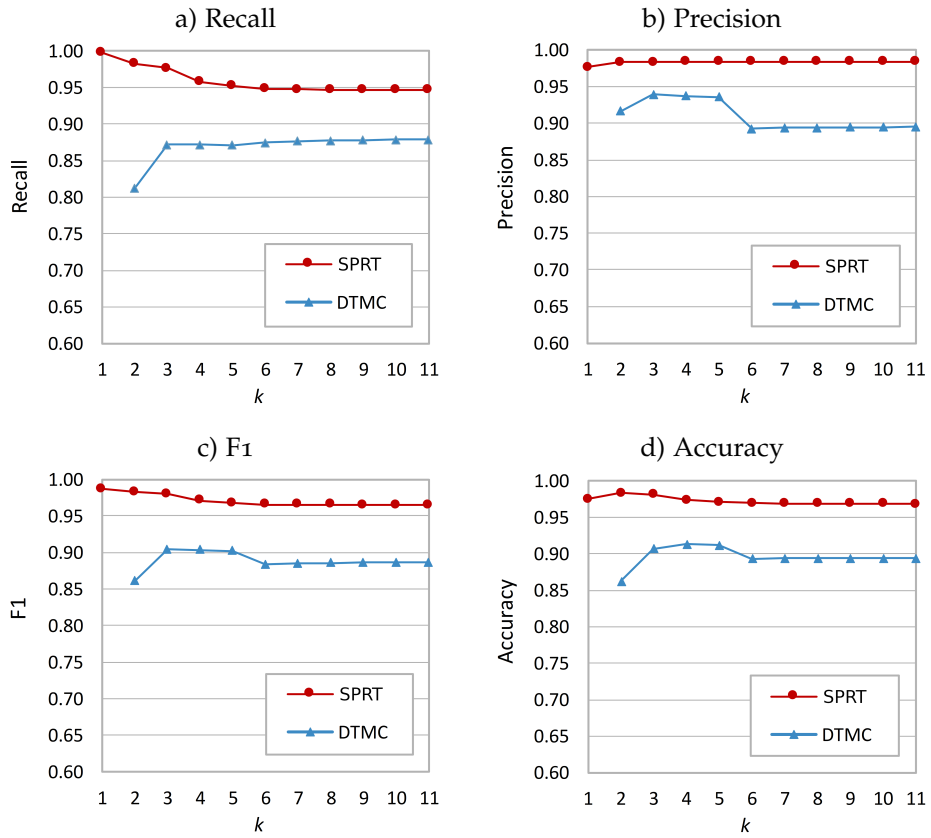
Figure 11 illustrates percentages of sessions classified by SPRT at each step, labeled according to the target classes. The plot shows that nearly all visitors identified at the first request, consisting in 15% of classified sessions, were bots, whereas at the second step 67% of visitors are correctly labeled, subdivided into 25% bots and 42% humans. Furthermore, the maximum number of requests to be observed before making a decision with SPRT was 19, as shown in Figure 10.

By contrast, the DTMC algorithm managed to classify only 75% of sessions within the sixth request, with no further improvements on the cumulative percentage of classified sessions, that doesn't benefit of additional observations (see Figure 12), even after fifty requests.

Most importantly, given $k_{\min} = 2$, the DTMC algorithm could not take any decisions at the first step and Figure 13 shows a number of classified sessions increasing constantly along with the number of request observed.

For both methods, the vast majority of *undecided* sessions were the extremely short ones, up to three requests long, and a detailed breakdown of their compositions points out a considerable preponderance of bots (see Figure 14 and Figure 15)

Figure 16: Cumulative performance scores vs decision steps (scenario 1 - excluding undecided sessions)



The plots in Figure 16 and Figure 17 respectively report the evolution of performance scores, incrementally obtained by considering true and false negatives and positives at each new observation, versus the number of steps that were sufficient to take a decision for both scenario 1 and scenario 2.

With regard to the quality of decisions, Table 8 summarizes the performance scores for both experimental scenarios.

Looking only at classified sessions, as of scenario 1, Table 8 presents very high efficiency for both methods, even if the proposed SPRT approach clearly outperforms DTMC being able to correctly identify 97% of web agents, compared to 89% of the referenced solution.

Moreover, a recall value of 0.95 confirms the ability of SPRT to well detect bots on-the-fly, while precision (0.98), which in our case measures the extent of humans erroneously identified as bots, is even higher than recall, indicating that there is no detriment on human visitors. For the sake of completeness, recall and precision of DTMC are set only to 0.88 and 0.90, respectively.

In scenario 2, undecided sessions were considered as undetected bots thus sensibly decreasing the performance scores of SPRT and dramatically worsening DTMC ones, except for precision that was not impacted in both approaches. Yet, SPRT was able to detect 93% of all bots on-the-fly, whereas the fraction of bots identified by DTMC was limited to 69%. Finally, the values of 96% achieved

Figure 17: Cumulative performance scores vs decision steps (scenario 2 - with undecided sessions counted as humans)

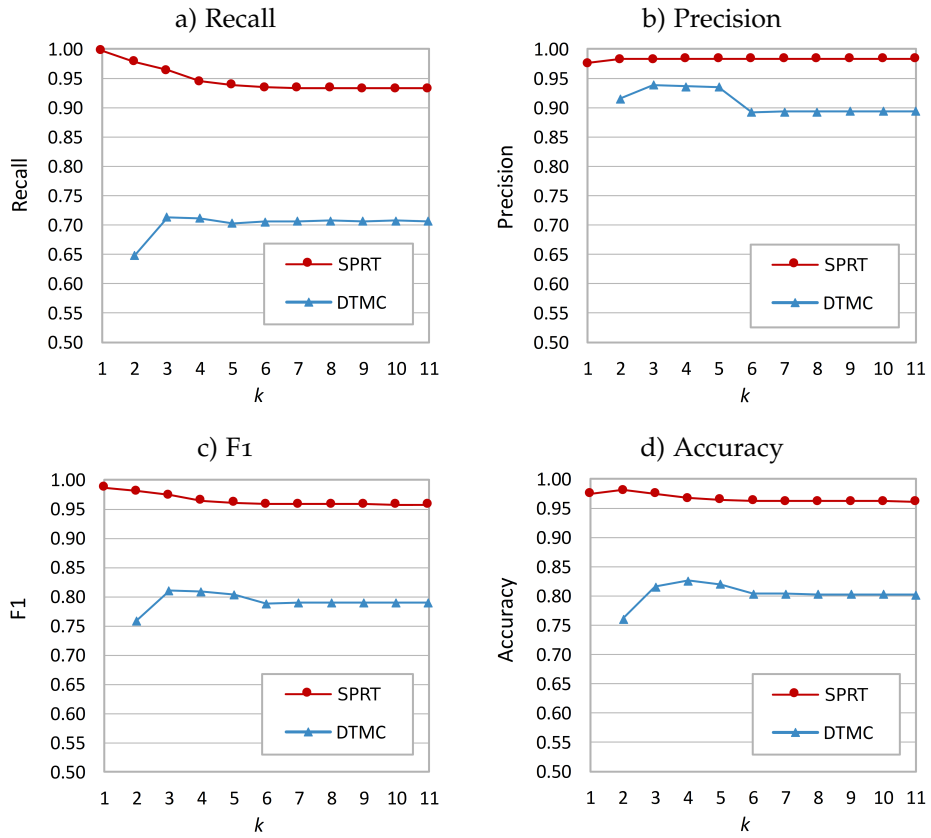


Table 8: Performance scores (with 10-fold cross-validation)

Evaluation scenario	**Metric (avg.)**	**SPRT**	**DTMC**
Scenario 1 (excluding undecided sessions)	Recall	0.95	0.88
	Precision	0.98	0.90
	F1	0.96	0.89
	Accuracy	0.97	0.89
Scenario 2 (including undecided sessions)	Recall	0.93	0.69
	Precision	0.98	0.90
	F1	0.96	0.78
	Accuracy	0.96	0.82

on F1 and accuracy with SPRT are far better than the relevant scores for DTMC, assigned to 78% and 82% respectively.

These considerations give evidence to the effectiveness of my approach for bot identification (SPRT). This works in a real-time interaction with a Web server in real-life operating conditions. In this case, the comparison algorithm either misclassifies the sessions or leaves them undecided. Basically, scenario 2

is more realistic than 1 because undecided sessions represent cases where the classifiers failed to detect bots, hence scores computed under these assumptions are more revealing of actual robustness of classifiers.

Figure 17 confirms the excellent efficiency of SPRT upon receiving the initial requests over the great majority of test sessions, which is the main goal of an early classification task. This is verified by F1 score, ranging from 0.96 to 0.98, that achieves the highest values at the very first requests, slightly decreasing as time goes by.

Observing the trend of recall, the highest values are achieved on the first three steps, with 0.998, 0.979, and 0.964, respectively, whereas precision always remains above 0.98. Anyhow, all performance scores for SPRT flatten at about the 7^{th} step, where most sessions have already been identified.

Same flattening behavior can be observed for DTMC on cumulative performance scores, in line with the conclusions drawn in [23], but at smaller values. The farthest decision step for SPRT was the 19^{th}, contrary to DTMC that not only had to analyze long sessions till their end, but also was not able to make a decision (Figure 17).

8.4.1 *Computational overhead*

From scalability point of view, an on-the-fly bot detection solution should be unobtrusive to the Web server and introduce only a minimal computational overhead to real-time processing of incoming requests.

The SPRT approach uses data available in HTTP request header without requiring any additional pre-processing, therefore it fosters its integration in server architectures.

From the computational complexity perspective, classification of a single request is performed in constant time. The neural network model, presented in Figure 5, has a fixed structure, allowing to compute the likelihoods $f_1(x_t)$ and $f_0(x_t)$, at a given time step t, in $O(1)$ time and space.

The incremental log-probability values $\log p_1(t)$ and $\log p_0(t)$ can be managed as session variables on the server therefore, for a session of length k, Sequential Probability Ratio Test can be estimated in at most $O(k)$ time. For N sessions the complexity does not significantly increase due to the possible parallelism offered by a multi-threaded implementation of the classifier.

Similar considerations apply to DTMC model, but the improved ability of SPRT in early classification with fewer observations, makes the expected computational overhead smaller.

9

Bot Detection with QEMC

9.1 The quantum early classifier module

An advantage of the proposed two-stage early classification model is the possible substitution of each stage independently of the other, providing that the communication interface remains the same. The Sequential Probability Ratio Test (SPRT) approach presented in the previous chapter is now replaced by Quantum-inspired Entangled Multinomial Classifier (QEMC), an innovative approach inspired to the principles of quantum mechanics.

In the present chapter, QEMC is applied to on-the-fly bot detection, deriving its application to the topic from the theoretical background presented in section 5.2 for the binary setting.

Being a binary problem, we can define an orthonormal basis to indicate the two target classes, namely $|0\rangle$ for humans and $|1\rangle$ for bots, as:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Let $\mathbf{x}_t$ be a sequence of HTTP request samples associated to a specific session and $y \in \{0, 1\}$ be the corresponding ground truth, which is obviously the same across each request in a session. The probability of request i being bot or human generated is estimated by means of the Multi-Layer Perceptron (MLP) at stage one and is stored in p_{ci}, where $c \in \{0, 1\}$ is the class label index.

As clarified earlier in chapter 5, quantum entanglement is a property that can be used to express a higher level of correlation among quantum states, therefore, as each request in a session belongs to a specific class across the whole sequence and they are reasonably correlated because they are generated by the same agent, it sounds sensible to hypothesize that quantum entanglement be capable of capturing and exposing the intrinsic correlation within each session, which is extremely difficult to model.

The probabilities of both classes, estimated by the Artificial Neural Network, can be used to build a quantum entangled representation of all subsequent requests in a session. The MLP classifier does not capture any temporal in-

formation but here it is used to assign the class likelihood of each individual sample.

Since the request order in each sequence is preserved to reflect the web navigation pattern, QEMC deals with correlation by means of entanglement.

As expressed by Equation 13, given the probabilities of the i-th observation in the sequence of length T, they can be related to the two basis states $|0\rangle$ and $|1\rangle$ by computing the coefficients α_i and β_i as

$$\alpha_i = \sqrt{p_{0i}} \qquad \text{and} \qquad \beta_i = \sqrt{p_{1i}} \tag{36}$$

and then create the T-qubits entangled states $|\psi_0\rangle$ and $|\psi_1\rangle$, according to Equation 15, from

$$\begin{cases} |\psi_0\rangle = \alpha\,|00...0\rangle = \alpha_0\,|0\rangle \otimes \alpha_1\,|0\rangle \otimes \cdots \otimes \alpha_{T-1}\,|0\rangle \\ |\psi_1\rangle = \beta\,|11...1\rangle = \beta_0\,|1\rangle \otimes \beta_1\,|1\rangle \otimes \cdots \otimes \beta_{T-1}\,|1\rangle \end{cases} \tag{37}$$

The state represented by a stream of T requests can be then expresses as the superposition of the two entangled states from Equation 37:

$$|\psi\rangle = |\psi_0\rangle + |\psi_1\rangle \tag{38}$$

In order to tell whether the current sample was generated by a bot or a human, it is necessary to measure, from the entangled state $|\psi\rangle$ by means of Equation 22 or Equation 25, the probabilities of the basis states $|0\rangle$ and $|1\rangle$ and compare those measurements against a properly tuned threshold C to take a decision, if enough information is contained in the given $|\psi\rangle$.

If no decision can be taken at current time step, then another observation is added to compute a new $|\psi\rangle$, until one of the measures meets the threshold or session ends, thus leaving the classification output as *undecided*.

The proposed approach works on normalized probabilities across the whole classification process, hence only one degree of freedom is sufficient to discriminate between classes. The *peep* mechanism introduced to manage memory space issues does not represent an actual degree of freedom because it is not related to data structure but rather to hardware specifications.

In order to experiment an additional degree of freedom, a variation of the main implementation of QEMC has been tested by computing the probability amplitudes, as of Equation 36, by means of

$$\alpha_i^{grade} \quad \text{with} \quad grade \in \mathbb{R}^+ \tag{39}$$

Even if α_i^{grade} cannot be considered a probability amplitude anymore, this option acts like a *fuzzyness* index, and it is beneficial to improve the classification results and tune the output of quantum classifier.

For instance, a proper value of *grade* applied in evaluation of optimal solution according to Pareto analysis allows to dramatically reduce the number of unclassified sessions. Moreover, when $grade = 0.5$, the solution is perfectly equivalent to the formal theory.

9.2 Experimental results and discussion

9.2.1 *The test scenarios*

The effectiveness of proposed method can be demonstrated with respect to the most representative performance metrics for the analyzed dataset and it is helpful to compare the algorithm with one that shows optimal results on the same problem.

In chapter 4, we showed that Sequential Probability Ratio Test from Wald [86] is able to outdo the allegedly best algorithm for real-time bot detection [23], therefore it has been compared with QEMC on the same probabilities previously used in the chapter mentioned above.

Presently, to the best of our knowledge, the SPRT method, proposed in [76], outperforms all other *state-of-the-art* approaches.

The main focus of the present work is not necessarily showing that the new approach perform better than *state-of-the-art* methods, but proving the effectiveness of a new paradigm that exploits quantum properties to take timely and reliable decisions.

The implemented two-stage model was beneficial to support the deployment of both Sequential Probability Ratio Test and Quantum-inspired Entangled Multinomial Classifier, along with the synoptical comparison of the respective results.

Three scenarios have been chosen to fairly and extensively compare the proposed and the reference approaches and possibly highlight any weaknesses in the new method, as visible in Table 9.

Table 9: Experimental scenarios

Scenario	Validation	peep	Lower Thr.	Upper Thr.	grade
A	50%	4	0.039	0.9	$0.2 \Rightarrow 2.6$
B	70%	4	$0.05 \Rightarrow 0.25$	$0.75 \Rightarrow 0.95$	0.5
C	70%	6	0.10	0.85	$0.1 \Rightarrow 2.6$

The number of sessions used for training has been gradually reduced down to the 30% of entire dataset and the *peep* and *grade* hyper-parameters are valid for the Quantum-inspired approach only.

Moreover, given that SPRT has been implemented by means of logarithmic expressions, the thresholds reported in Table 9 are transparently converted into their log equivalent.

It is worth noting that the *peep* mechanism, although required to control the computational impact, represents a disadvantage for QEMC algorithm because it bounds the method's *look-back* memory.

For each scenario described in Table 9, the same performance indicators have been considered:

- LDS: length of the decision sequence; low is good;

- ACC: accuracy of classification, defined as the total number of correct assignments divided by the total number of sessions; high is good;
- TUC: total number of unclassified sessions left; to be minimized

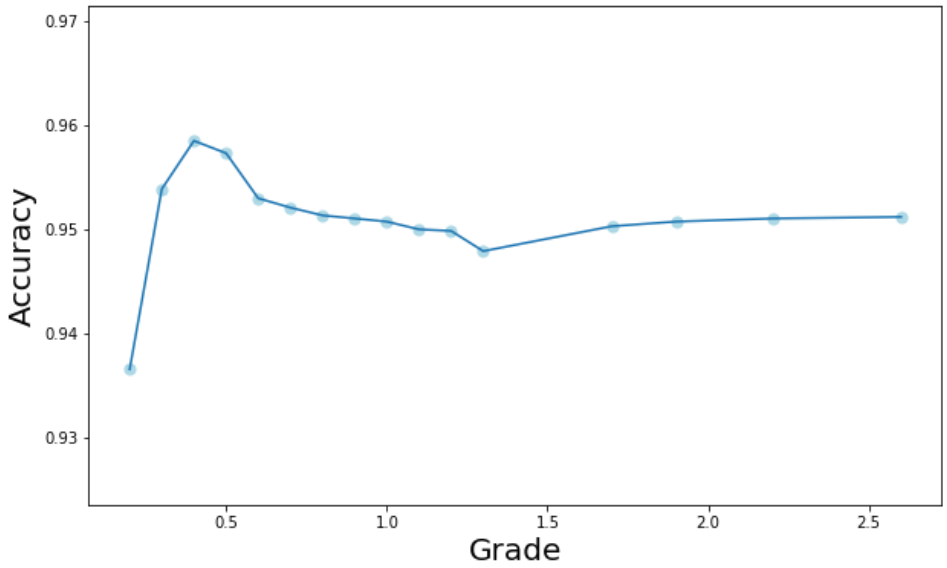
These metrics represent three objective functions that ideally should be optimized simultaneously for proper decision making. Pareto front plots have been used to balance these contrasting goals because our aim is accuracy maximization against minimization of decision steps and unclassified sessions. This means that multiple solution can optimize the desired objectives and every *non-dominated* solutions is Pareto optimal (see subsubsection 8.1.0.1) and represents an acceptable solution to the problem.

In the present work, only extrema are considered as they represent the best solution on at least one observable metric.

9.2.2 *Scenario A - classification accuracy vs grade*

This scenario has been setup to assess the impact of variable values of *grade* on the performance indicators. Since, this hyper-parameter only affects the outcome of quantum-inspired approach, the results of SPRT are constant in the comparison. Specifically, classification on SPRT ends with ACC equal to 0.9422, leaving only 4 unclassified sessions and using 3 steps for LDS.

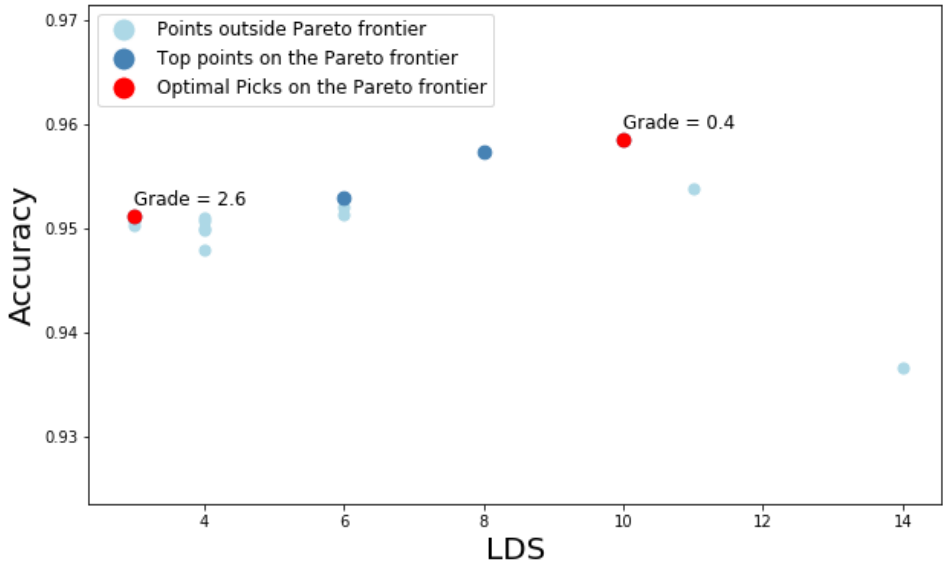
Figure 18: Scenario A - Accuracy vs Grade.



The decision thresholds have been set to fixed values, identified as optimal by means of Pareto analysis, and 50% of available sessions have been set aside for model validation.

Concerning QEMC, different values of *grade* have been tested, as shown in Figure 18 but, according to the Pareto frontier plot in Figure 19, the optimal points to consider for the comparison with SPRT correspond to grade 0.4, which

Figure 19: Scenario A - Pareto front analysis.



maximizes the accuracy, and 2.6 which minimizes the length of decision sequence to the same value as SPRT.

At *grade* 0.4, ACC value is 0.9585, the highest for this setting, but the number of unclassified sessions is 50, which is extremely high compared to SPRT, and LDS is 10.

Conversely, at *grade* 2.6, accuracy is only slightly less than in the previous case (ACC=0.9512 with a variation $\Delta = -0.0073$) but LDS is exactly the same as in SPRT and the number of unclassified sessions drops to zero. Nevertheless, in both cases, classification accuracy is greater QEMC (worst case variation has $\Delta = 0.009$).

9.2.3 *Scenario B - dependence on thresholds*

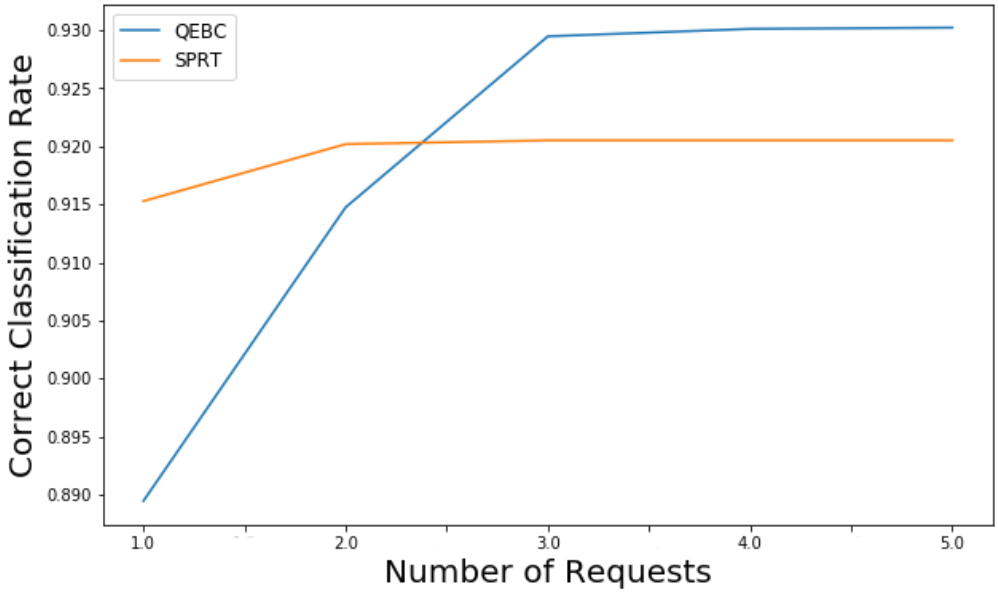
This scenario evaluates the classification results with regard to variable threshold values on 70% of sessions used for validation with *grade* set at 0.5, which is the default value for QEMC.

In order to make the two approaches comparable with regard to hyper-parameters selection, a modified version of QEMC binary classifier was tested, adding a selection threshold for eventually restricting the decision options. In this case the class label at request k is assigned by changing Equation 30 as follows:

$$dec_k = \begin{cases} 1 & \text{if} \quad p_i(k) \geq T_1 \\ 0 & \text{if} \quad p_i(k) \leq T_0 \\ \oslash & \text{if} \quad T_0 < p_i(k) < T_1. \end{cases} \tag{40}$$

The best results for SPRT were achieved with lower and upper thresholds set to the logarithm of 0.1 and 0.85 respectively; in this configuration, ACC is

Figure 20: Scenario B - Accuracy vs Decision Step



0.9205, TUC is 8 and LDS is 3. The metrics for QEMC at the same thresholds values are slightly better in accuracy (0.9302), which means that the overall number of correctly classified sessions is greater, but it might take longer to make a decision (LDS = 5), even if in both cases the 90% of sessions is classified at the first step, and the number of unclassified sessions is almost doubled (TUC = 15).

The best threshold pair for QEMC is 0.25 for the lower and 0.95 for the upper threshold where, despite even greater values of LDS (7) and TUC (23), the accuracy sensibly rises to 0.9527 ($\Delta = +0.0322$ versus best SPRT) and the 90% of sessions is classified within the second step. For these threshold values, the accuracy of SPRT is slightly lower (0.9204) than the best case, but the number of unclassified sessions decreases to 4 while maintaining the same LDS value. Despite the lower number of undecided sessions, the lower accuracy value indicates that SPRT has actually a greater number of misclassified sessions.

For the current setting, Figure 20 visualizes the rate of correctly classified sessions for the two methods: SPRT identifies a greater percentage at the first two requests but no great improvement is achieved on the third and last step. Conversely, QEMC takes over at the third request and the overall performance is nearly 1% greater than the reference method.

9.2.4 *Scenario C - variable grade with larger peep*

The third scenario compares the performance indicators when varying *grade* at constant thresholds, optimal for SPRT, and with $peep = 6$, which is expected to improve accuracy of QEMC by considering longer sub-sequences in the decision process.

The configuration of current scenario is unchanged from the previous one, with a non negligible reduction in the training dataset with regard to scenario A. Since the *peep* mechanism only applies to the quantum-inspired approach, the results for the probabilistic method remain unaltered with $ACC = 0.9205$, $TUC = 8$ and $LDS = 3$, while conversely the expedient improves QEMC scores at the Pareto optimal values of *grade*.

Figure 21: Scenario C - Accuracy vs Grade.

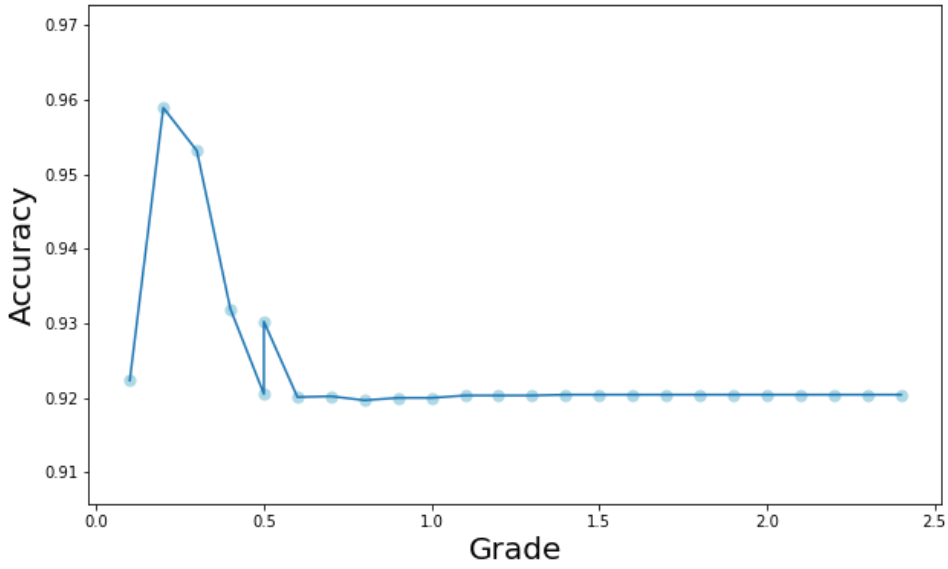
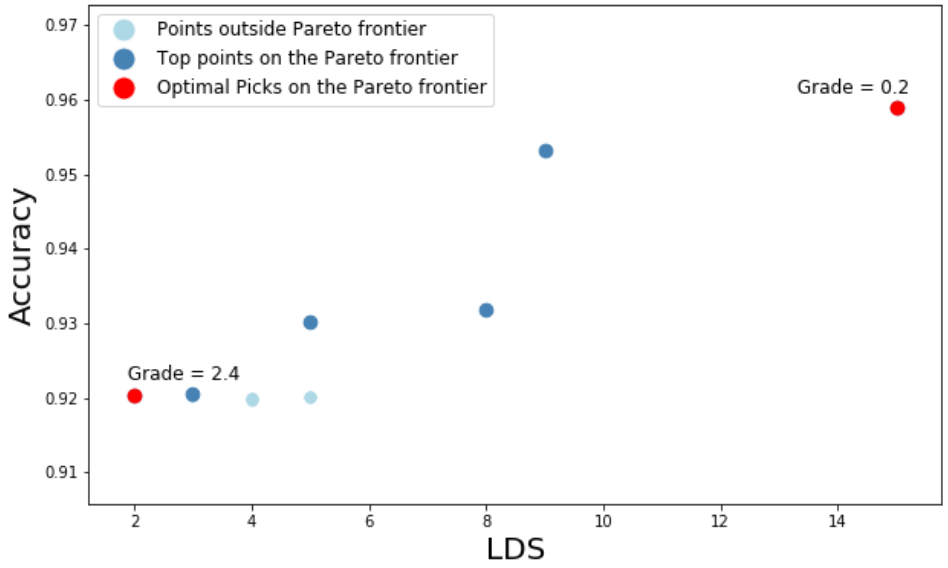


Figure 22: Scenario C - Pareto front analysis.



The optimal *grade* value to maximize accuracy is 0.2, as shown in 22, where accuracy is 0.9589, a bit higher ($\Delta = +0.0004$) than in Scenario A with *peep* at 4, showing that it is possible to achieve better classification rates by considering

more samples at the expense of computational effort. This is also paid for in terms of LDS, that grows up to 15, TUC that spikes to 91 and on the number of steps required to classify the 90% of the sessions which becomes 3.

On the other side, the optimal value of *grade* to minimize LDS is 2.4, which not only requires at most 2 samples to take a reliable decision but also allows to achieve zero on the total number of unclassified sessions. The good point here is that accuracy is only 10^{-4} worse than for SPRT, with only 1 request needed to classify 90% of the sessions in both cases.

The three scenarios proposed above are representative of the various combinations of post-training hyper-parameters and expose both the pros and cons of the novel quantum-inspired approach.

Classification accuracy for Quantum-inspired Entangled Multinomial Classifier can be sensibly boosted by properly selecting the *peep* and *grade* values, at the same threshold conditions, by means of Pareto analysis. Moreover, the same parameters can be tuned to target some specific objectives, such as zero unclassified sessions or a shorter decision sequence, while preserving the performance indicators that, in the worst case, are fairly equal to those from Sequential Probability Ratio Test. In fact, by adjusting *peep* and *grade*, it is possible to increase the convergence speed of the classification algorithm and reduce the number of requests needed to take a decision to an even smaller value than SPRT.

It is worth noting that a reduction in the training size of the dataset has a minimal impact on classification accuracy for quantum-inspired algorithm, in the same testing conditions.

Experimental evidence shows that, with a validation ratio of 50%, accuracy is 0.9573 for QEMC and 0.9421 for SPRT whereas, when 30% of the sessions is used for training, the corresponding values are 0.9535 and 0.9204 respectively. Hence $\Delta_{QEMC} = -0.0038$ and $\Delta_{SPRT} = -0.0217$, which is nearly 6 times greater than the former.

Another important consideration is related to the *peep* value: the adoption of such mechanism is imposed by the computational performance downgrade on long sequences when the decision process requires to consider many requests to meet the desired confidence level.

However, regardless of the length of a session, the number of samples that have to be taken into account, as shown in section 6.4 and section 9.2, it is often limited to 4 to 6 samples. Greater *peep* values do not bring any benefit to the classification performance but increase the computational effort, making the approach less suitable for a real-time application.

Nonetheless, being bot detection a binary problem, it was possible to confirm the above assertion by a modified implementation software that relies on simplified math instead of the tensor product required in the multi-class setting. This alternative solution can handle sequences of any length without going to the *peep* workaround. Results from such experiments guarantee the same exact level of accuracy achieved with tensor implementation, even if at a sensibly higher speed: unfortunately in the multinomial setting it is not possible to devise an alternative lightweight computational approach.

Finally, while SPRT is designed as a binary classifier and requires a modified approach to be applied in a multi-class problem, the QEMC method is natively suited for multinomial problems by simply expanding the orthonormal basis through the addition of further basis states.

10

Conclusions and Final Remarks

The present thesis is the result of my PhD studies and research activities, that started with a very specific research question, such as on-the-fly bot detection from HTTP request logs.

As my understanding of the task was progressing, my interest in finding a generalized approach that could solve similar problems also increased in parallel.

After analyzing the state-of-the-art methods and implementing an effective solution based on Sequential Probability Ratio Test, the growing attention of the Computer Science community towards quantum-computing and its applications to solve complex problems, made me wonder if it would be possible to take advantage of its underlying principles in time series classification.

The study of quantum mechanics properties and qubit simulation was the first step into developing the proposed quantum-inspired classifier, termed Quantum-inspired Entangled Multinomial Classifier (QEMC), that was initially designed into the bot detection binary setting and subsequently extended into a multi-class version, with reject option, for more generic purposes.

In the present thesis, the general structure of a temporal sequence of data was analyzed, highlighting the benefits of real time classification for stationary and non-stationary data streams.

In the chapter 8, a novel method for Web bot detection on a Web server in real time was proposed, based on a two-stage classifier that combines a neural network model and the Sequential Probability Ratio Test to classify an active visitor as a bot or human as early as possible.

The extensive experimental study, tested on traffic streams from an actual Polish server, showed that SPRT is able to detect as much as 93% of all bots on-the-fly and is especially powerful given a very limited number of observations.

Contrarily to other bot detection approaches [6, 23, 80], SPRT does not set a minimum number of HTTP requests to be observed before allowing for a decision. Moreover, 0.98 in precision asserts that very few human visitors are erroneously classified as bots: this is a very important aspect for transparency to human users of bot detection mechanisms.

In fact, the Sequential Probability Ratio Test classifier can be implemented as a lightweight extension to Web server software or integrated with other similar tools, like CAPTCHA.

The theoretical formulation of the algorithm states that it is possible to estimate the number of observations required to achieve a given error rate, but current implementation does not support this feature, which could be included in future research works.

As a different approach from SPRT, an innovative quantum-inspired multinomial classifier for early detection of significant events on time series, that has been validated both in a synthetic experimental setting and in the bot detection application.

The present study confirms that the proposed technique, which relies on superposition and entanglement to integrate the class probabilities, estimated by an upstream stage, of every collected event in a time series, can produce an overall score capable of supporting trustful decisions even in case of a limited number of events, both in the binary and in the multinomial setting.

The effectiveness of this method has also been successfully compared with other approaches and specifically with SPRT on bot detection, algorithm that was proven to outperform some consolidated state-of-the-art techniques.

Similarly, its results were analyzed with reference to the contrasting objectives of classification accuracy, number of undecided sessions and speed of decision, showing that the proposed quantum-inspired algorithm, in my opinion, natively covers an area of application (non-stationary data stream classification) that so far has not yet found reliable and performing approaches.

With regard to the methods analyzed in chapter 3, some additional notes are worth reporting:

1. QEMC is tolerant against non-standardized numerical features, which is usually considered a compelling transformation for machine learning tasks;

2. with QEMC, it is possible to dramatically reduce the number of training sequences with no significant drop of classification scores;

3. in current configuration of the classification framework, solutions are not interpretable, therefore some areas of application might be precluded to QEMC;

4. no estimate on reliability of decisions is currently available in QEMC;

5. dependencies on *grade* parameter have not been explored in depth, but could open the way to a *fuzzy* flavor of the classifier.

In my opinion, the last three items represent interesting areas of investigation, where near future research should be directed.

Replacement of the Artificial Neural Network with explainable ways to compute the probability estimates of observations might open new perspectives for the quantum-inspired technique, especially if accompanied by a measure of decision reliability.

Last but not least, we are starting to apply QEMC to other industrial sectors, such as automotive, and to lay the basis for future research activities on multinomial time series classification.

PART III

Appendix

11

Appendix on Wald's Theorems

The present appendix provides the mathematical proof of the two theorems from Wald that are cited in chapter 4, with implicit reference to the symbols and formalism used therein.

11.1 Proof of equation 10

Theorem: A is upper bounded by $\frac{1-\beta}{\alpha}$ and B is lower bounded by $\frac{\beta}{1-\alpha}$
Proof
For each sample $\{x_1, \ldots, x_m\}$ such that $dec_k = 1$, as of Equation 8, holds

$$p(x_1, \ldots, x_m | y = 1) \geq p(x_1, \ldots, x_m | y = 0)$$

Since this equation is valid for all samples assigned to class 1, for the probabilities holds

$$P(dec = 1 | y = 1) \geq P(dec = 1 | y = 0)$$

The left term represents the probability of correctly labeling a sequence in class 1, that is $1 - \beta$, whereas the right term is the probability of incorrect labeling for class 0, which is α.

Hence we can write $1 - \beta \geq A \cdot \alpha$, which leads to

$$A \leq \frac{1-\beta}{\alpha} \tag{41}$$

A similar reasoning can be applied to the lower bound, which proves that $B \geq \frac{\beta}{1-\alpha}$.

11.2 Proof of equation 11

Theorem: When $A^\star$ and $B^\star$ are selected instead of optimal values A and B, for the real error probabilities $\alpha^\star$ and $\beta^\star$ holds that $\alpha^\star + \beta^\star \leq \alpha + \beta$

Proof
When assigning the values of $A^\star$ and $B^\star$ as of Equation 10, we can write

$$\frac{\alpha^\star}{1-\beta^\star} \leq \frac{1}{A^\star} = \frac{\alpha}{1-\beta}$$

Multiplying both sides by $(1-\beta^\star)\cdot(1-\beta)$, we obtain

$$\alpha^\star \cdot (1-\beta) \leq \alpha \cdot (1-\beta^\star)$$

whence we can write

$$\alpha^\star - \alpha^\star\beta \leq \alpha - \alpha\beta^\star \tag{42}$$

Similarly, from

$$\frac{\beta^\star}{1-\alpha^\star} \leq B^\star = \frac{\beta}{1-\alpha}$$

after multiplying both sides by $(1-\alpha^\star)\cdot(1-\alpha)$, we have

$$\beta^\star - \alpha\beta^\star \leq \beta - \alpha^\star\beta \tag{43}$$

Finally, summing together the inequalities in 42 and 43, we obtain the expected result

$$\alpha^\star + \beta^\star \leq \alpha + \beta \tag{44}$$

Bibliography

[1] Dilara Acarali, Muttukrishnan Rajarajan, Nikos Komninos and Ian Herwono. 'Survey of approaches and features for the identification of HTTP-based botnet traffic'. In: Journal of Network and Computer Applications 76 (2016), pp. 1–15. issn: 1084-8045. doi: 10.1016/j.jnca.2016.10.007.

[2] Ratnadip Adhikari and Ramesh K Agrawal. 'An introductory study on time series modeling and forecasting'. In: arXiv preprint arXiv:1302.6613 (2013).

[3] Shafiq Alam, Gillian Dobbie, Yun Sing Koh and Patricia Riddle. 'Web bots detection using particle swarm optimization based clustering'. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC'14). IEEE. 2014, pp. 2955–2962.

[4] Arnold O Allen and Statistics Probability. Queueing Theory with Computer Science Applications. 1990.

[5] Hyrum S Anderson, Nathan Parrish and Maya R Gupta. Early Time Series Classification with Reliability Guarantee. Tech. rep. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2012.

[6] Andoena Balla, Athena Stassopoulou and Marios D. Dikaiakos. 'Real-time Web crawler detection'. In: Proceedings of the 18th International Conference on Telecommunications. May 2011, pp. 428–432. doi: 10.1109/CTS.2011.5898963.

[7] ZoltáN Bankó and JáNos Abonyi. 'Correlation based dynamic time warping of multivariate time series'. In: Expert Systems with Applications 39.17 (2012), pp. 12814–12823.

[8] Zoltán Bankó and János Abonyi. 'Correlation Based Dynamic Time Warping'. In: 8th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics, CINTI 2007. 2007, pp. 295–306.

[9] Basic quantum circuit simulation in Python. http://jarrodmcclean.com/basic-quantum-circuit-simulation-in-python/. Accessed: 2019-07-29.

[10] Yoshua Bengio, Ian Goodfellow and Aaron Courville. Deep learning. Vol. 1. Citeseer, 2017.

[11] T. Berners-Lee, R. Fielding and H. Frystyk. Hypertext Transfer Protocol – HTTP/1.0. IETF RFC 1945. May 1996.

[12] Francesca Biagini and Massimo Campanino. 'Discrete Time Markov Chains'. In: Elements of Probability and Statistics: An Introduction to Probability with de Finetti's Approach and to Bayesian Statistics. Cham: Springer International Publishing, 2016, pp. 81–87. isbn: 978-3-319-07254-8.

[13] Christian Bomhardt, Wolfgang Gaul and Lars Schmidt-Thieme. 'Web robot detection – preprocessing Web logfiles for robot detection'. In: New Developments in Classification and Data Analysis. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 113–124. isbn: 978-3-540-27373-8.

[14] George EP Box, Gwilym M Jenkins, Gregory C Reinsel and Greta M Ljung. Time series analysis: forecasting and control. John Wiley & Sons, 2015. isbn: 978-1-118-67502-1.

[15] Peter J Brockwell and Richard A Davis. Time series: theory and methods. Springer Science & Business Media, 2013.

[16] Peter J Brockwell, Richard A Davis and Matthew V Calder. Introduction to time series and forecasting. Vol. 2. Springer, 2002.

[17] C. K. Chow. 'An optimum character recognition system using decision functions'. In: IRE Transactions on Electronic Computers EC-6.4 (Dec. 1957), pp. 247–254. issn: 0367-9950. doi: 10.1109/TEC.1957.5222035.

[18] Zi Chu, Steven Gianvecchio, Aaron Koehl, Haining Wang and Sushil Jajodia. 'Blog or Block: Detecting Blog Bots Through Behavioral Biometrics'. In: Computer Networks 57.3 (Feb. 2013), pp. 634–646. issn: 1389-1286. doi: 10.1016/j.comnet.2012.10.005.

[19] Asma Dachraoui, Alexis Bondu and Antoine Cornuéjols. 'Early classification of time series as a non myopic sequential decision making problem'. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer. 2015, pp. 433–447.

[20] Marco Di Marzio and Charles C Taylor. 'Kernel density classification and boosting: an L 2 analysis'. In: Statistics and Computing 15.2 (2005), pp. 113–123.

[21] Marios D. Dikaiakos, Athena Stassopoulou and Loizos Papageorgiou. 'An Investigation of Web Crawler Behavior: Characterization and Metrics'. In: Comput. Commun. 28.8 (May 2005), pp. 880–897. issn: 0140-3664. doi: 10.1016/j.comcom.2005.01.003.

[22] Derek Doran and Swapna S. Gokhale. 'Web robot detection techniques: overview and limitations'. In: Data Mining and Knowledge Discovery 22 (June 2011), pp. 183–210.

[23] Derek Doran and Swapna S. Gokhale. 'An integrated method for real time and offline web robot detection'. In: Expert Systems 33.6 (2016), pp. 592–606. issn: 1468-0394.

[24] Artur Ekert, PM Hayden and Hitoshi Inamori. 'Basic concepts in quantum computation'. In: Coherent atomic matter waves. Springer, 2001, pp. 661–701.

[25] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. IETF RFC 2616. June 1999.

[26] Jerome Friedman, Trevor Hastie and Robert Tibshirani. The elements of statistical learning. Vol. 1. 10. Springer series in statistics New York, 2001.

[27] Vladimir Geroimenko. Dictionary of XML Technologies and the Semantic Web. Springer-Verlag, London, UK, 2004.

[28] Mohamed F Ghalwash and Zoran Obradovic. 'Early classification of multivariate temporal observations by extraction of interpretable shapelets'. In: BMC bioinformatics 13.1 (2012), p. 195.

[29] Mohamed F Ghalwash, Vladan Radosavljevic and Zoran Obradovic. 'Utilizing temporal patterns for estimating uncertainty in interpretable early decision making'. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM. 2014, pp. 402–411.

[30] Mohamed F Ghalwash, Dušan Ramljak and Zoran Obradović. 'Early classification of multivariate time series using a hybrid HMM/SVM model'. In: 2012 IEEE International Conference on Bioinformatics and Biomedicine. IEEE. 2012, pp. 1–6.

[31] S. Gianvecchio, M. Xie, Z. Wu and H. Wang. 'Humans and bots in Internet chat: Measurement, analysis, and automated classification'. In: IEEE ACM T. Network. 19.5 (2011), pp. 1557–1571.

[32] Toni Gržinić, Leo Mršić and Josip Šaban. 'Lino - An Intelligent System for Detecting Malicious Web-Robots'. In: Intelligent Information and Database Systems. Cham: Springer International Publishing, 2015, pp. 559–568. isbn: 978-3-319-15705-4.

[33] Elloá B Guedes, Francisco Marcos de Assis and Rex AC Medeiros. 'Fundamentals of Quantum Information Processing'. In: Quantum Zero-Error Information Theory. Springer, 2016, pp. 7–26.

[34] Weigang Guo, Shiguang Ju and Yi Gu. 'Web robot detection techniques based on statistics of their requested URL resources'. In: Proceedings of the 9th International Conference on Computer Supported Cooperative Work in Design. Vol. 1. May 2005, pp. 302–306. doi: 10.1109/CSCWD.2005.194187.

[35] Javad Hamidzadeh, Mahdieh Zabihimayvan and Reza Sadeghi. 'Detection of Web site visitors based on fuzzy rough sets'. In: Soft Computing 22.7 (Apr. 2018), pp. 2175–2188. issn: 1433-7479. doi: 10.1007/s00500-016-2476-4.

[36] N. Hatami and C. Chira. 'Classifiers with a reject option for early time-series classification'. In: 2013 IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL). Apr. 2013, pp. 9–16.

[37] Pedram Hayati, Vidyasagar Potdar, Kevin Chai and Alex Talevski. 'Web Spambot Detection Based on Web Navigation Behaviour'. In: Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA'10). Apr. 2010, pp. 797–803. doi: 10.1109/AINA.2010.92.

[38] Guoliang He, Yong Duan, Rong Peng, Xiaoyuan Jing, Tieyun Qian and Lingling Wang. 'Early classification on multivariate time series'. In: Neurocomputing 149 (2015), pp. 777–787.

[39] Carl W. Helstrom. Quantum detection and estimation theory. Mathematics in science and engineering v. 123. New York: Academic Press, 1976. isbn: 978-0-12-340050-5.

[40] Ching-Hsiang Hsu, Chun-Ying Huang and Kuan-Ta Chen. 'Fast-Flux Bot Detection in Real Time'. In: Recent Advances in Intrusion Detection. RAID'10. Ed. by Somesh Jha, Robin Sommer and Christian Kreibich. Berlin, Heidelberg: Springer, 2010, pp. 464–483. isbn: 978-3-642-15512-3. doi: 10.1007/978-3-642-15512-3_24.

[41] John D Hunter. 'Matplotlib: A 2D graphics environment'. In: Computing in science & engineering 9.3 (2007), pp. 90–95.

[42] Gregoire Jacob, Engin Kirda, Christopher Kruegel and Giovanni Vigna. 'PUBCRAWL: Protecting Users and Businesses from CRAWLers'. In: Proceedings of the 21st USENIX Conference on Security Symposium. Security'12. Bellevue, WA: USENIX Association, 2012, pp. 25–25.

[43] Agnieszka Jakóbik, Francesco Palmieri and Joanna Kołodziej. 'Stackelberg games for modeling defense scenarios against cloud security threats'. In: Journal of Network and Computer Applications 110 (2018), pp. 99–107. issn: 1084-8045. doi: 10.1016/j.jnca.2018.02.015.

[44] D. Kristol and L. Montulli. HTTP State management mechanism. IETF RFC 2109. Feb. 1997.

[45] Shinil Kwon, Young-Gab Kim and Sungdeok Cha. 'Web Robot Detection Based on Pattern-matching Technique'. In: Journal of Information Science 38.2 (Apr. 2012), pp. 118–126. issn: 0165-5515. doi: 10.1177/0165551511435969.

[46] Shinil Kwon, Myeongjin Oh, Dukyun Kim, Junsup Lee, Young-Gab Kim and Sungdeok Cha. 'Web robot detection based on monotonous behavior'. In: Proceedings of the information science and industrial applications 4 (2012), pp. 43–48.

[47] Srivatsan Laxman and P Shanti Sastry. 'A survey of temporal data mining'. In: Sadhana 31.2 (2006), pp. 173–198.

[48] Junsup Lee, Sungdeok Cha, Dongkun Lee and Hyungkyu Lee. 'Classification of Web Robots: An Empirical Study Based on over One Billion Requests'. In: Comput. Secur. 28.8 (Nov. 2009), pp. 795–802. issn: 0167-4048. doi: 10.1016/j.cose.2009.05.004.

[49] Wei-Zhou Lu and Shun-Zheng Yu. 'Web Robot Detection Based on Hidden Markov Model'. In: Proceedings of International Conference on Communications, Circuits and Systems. Vol. 3. 2006, pp. 1806–1810. doi: 10.1109/ICCCAS.2006.285024.

[50] Wes McKinney et al. 'Data structures for statistical computing in python'. In: Proceedings of the 9th Python in Science Conference. Vol. 445. Austin, TX. 2010, pp. 51–56.

[51] Kaisa Miettinen. Nonlinear Multiobjective Optimization. Boston: Kluwer Academic Publishers, 1999.

[52] Vicente Moret-Bonillo. 'Can artificial intelligence benefit from quantum computing?' In: Progress in Artificial Intelligence 3.2 (2015), pp. 89–105.

[53] Usue Mori, Alexander Mendiburu, Eamonn Keogh and Jose A Lozano. 'Reliable early classification of time series based on discriminating the classes over time'. In: Data mining and knowledge discovery 31.1 (2017), pp. 233–263.

[54] Travis E Oliphant. A guide to NumPy. Vol. 1. Trelgol Publishing USA, 2006.

[55] Randal S Olson, William La Cava, Patryk Orzechowski, Ryan J Urbanowicz and Jason H Moore. 'PMLB: a large benchmark suite for machine learning evaluation and comparison'. In: BioData mining 10.1 (2017), p. 36.

[56] Vilfredo Pareto. Manuel d'économie politique. 2014.

[57] Nathan Parrish, Hyrum S Anderson, Maya R Gupta and Dun Yu Hsiao. 'Classifying with confidence from incomplete information'. In: The Journal of Machine Learning Research 14.1 (2013), pp. 3561–3589.

[58] F. Pedregosa et al. 'Scikit-learn: Machine Learning in Python'. In: Journal of Machine Learning Research 12 (2011), pp. 2825–2830.

[59] Goran Peskir and Albert Shiryaev. Optimal stopping and free-boundary problems. Springer, 2006.

[60] Nicolas Privault. Understanding Markov Chains: Examples and Applications. 2018.

[61] Qubits, Quantum Mechanics, and Computers. `http://www-inst.eecs.berkeley.edu/~{}cs191/fa14/`. Accessed: 2018-02-06, refer also to past years lectures.

[62] Lawrence R Rabiner. 'A tutorial on hidden Markov models and selected applications in speech recognition'. In: Proceedings of the IEEE 77.2 (1989), pp. 257–286.

[63] Mahmudur Rahman, Mizanur Rahman, Bogdan Carbunar and Duen Horng Chau. 'Search Rank Fraud and Malware Detection in Google Play'. In: IEEE Transactions on Knowledge and Data Engineering 29.6 (June 2017), pp. 1329–1342. issn: 1041-4347. doi: `10.1109/TKDE.2017.2667658`.

[64] Eleanor Rieffel and Wolfgang Polak. 'Quantum computing'. In: The Handbook of Technology Management 3 (2011).

[65] Stefano Rovetta, Alberto Cabri, Francesco Masulli and Grażyna Suchacka. 'Bot or Not? A Case Study on Bot Recognition from Web Session Logs'. In: Italian Workshop on Neural Nets. Springer. 2017, pp. 197–206.

[66] Tiago Santos and Roman Kern. 'A Literature Survey of Early Time Series Classification and Deep Learning'. In: SamI40 workshop at i-KNOW'16. 2016.

[67] Christian Hadiwijaya Saputra, Erwin Adi and Shintia Revina. 'Comparison of Classification Algorithms to tell Bots and Humans Apart'. In: Journal of Next Generation Information Technology 4 (Sept. 2013), pp. 23–32.

[68] Cédric Saule and Robert Giegerich. 'Pareto optimization in algebraic dynamic programming'. In: Algorithms for Molecular Biology 10.1 (July 2015), p. 22. doi: 10.1186/s13015-015-0051-7.

[69] Giuseppe Sergioli, Gustavo Martin Bosyk, Enrica Santucci and Roberto Giuntini. 'A Quantum-inspired Version of the Classification Problem'. en. In: International Journal of Theoretical Physics 56.12 (Dec. 2017), pp. 3880–3888. doi: 10.1007/s10773-017-3371-1.

[70] Giuseppe Sergioli, Roberto Giuntini and Hector Freytes. 'A new quantum approach to binary classification'. en. In: PLOS ONE 14.5 (May 2019). Ed. by Austin Lund. doi: 10.1371/journal.pone.0216224.

[71] Dilip Singh Sisodia, Shrish Verma and Om Prakash Vyas. 'Agglomerative Approach for Identification and Elimination of Web Robots from Web Server Logs to Extract Knowledge about Actual Visitors'. In: Journal of Data Analysis and Information Processing 03 (Jan. 2015), pp. 1–10. doi: 10.4236/jdaip.2015.31001.

[72] Athena Stassopoulou and Marios D. Dikaiakos. 'Web robot detection: a probabilistic reasoning approach'. In: Comput. Netw. 53.3 (Feb. 2009), pp. 265–278. issn: 1389-1286. doi: 10.1016/j.comnet.2008.09.021.

[73] Dusan Stevanovic, Aijun An and Natalija Vlajic. 'Feature evaluation for Web crawler detection with data mining techniques'. In: Expert Syst. Appl. 39.10 (Aug. 2012), pp. 8707–8717. issn: 0957-4174. doi: 10.1016/j.eswa.2012.01.210.

[74] Dusan Stevanovic, Natalija Vlajic and Aijun An. 'Detection of Malicious and Non-malicious Website Visitors Using Unsupervised Neural Network Learning'. In: Appl. Soft Comput. 13.1 (Jan. 2013), pp. 698–708. issn: 1568-4946. doi: 10.1016/j.asoc.2012.08.028.

[75] Emma Strubell. 'An introduction to quantum algorithms'. In: COS498 Chawathe Spring 13 (2011), p. 19.

[76] G. Suchacka, A. Cabri, S. Rovetta and F. Masulli. Efficient On-the-fly Web Bot Detection. IEEE Transactions on Knowledge and Data Engineering. Under review. 2019.

[77] Grażyna Suchacka. 'Analysis of aggregated bot and human traffic on e-commerce site'. In: Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on. IEEE. 2014, pp. 1123–1130.

[78] Grażyna Suchacka and Igor Motyka. 'Efficiency analysis of resource request patterns in classification of Web robots and humans'. In: Proceedings of the 32nd European Conference on Modelling and Simulation. 2018.

[79] Grażyna Suchacka and Mariusz Sobkow. 'Detection of Internet robots using a Bayesian approach'. In: 2015 IEEE 2nd International Conference on Cybernetics (CYBCONF). IEEE. 2015, pp. 365–370.

[80] Pang-Ning Tan and Vipin Kumar. 'Discovery of Web robot sessions based on their navigational patterns'. In: Data Min. Knowl. Discov. 6.1 (Jan. 2002), pp. 9–35. issn: 1384-5810. doi: 10.1023/A:1013228602957.

[81] Prayag Tiwari and Massimo Melucci. 'Towards a Quantum-Inspired Binary Classifier'. In: IEEE Access 7 (2019), pp. 42354–42372. issn: 2169-3536. doi: 10.1109/ACCESS.2019.2904624. url: https://ieeexplore.ieee.org/document/8671690/ (visited on 18/09/2019).

[82] Udger. url: https://udger.com%20(access%20date:%20September%204,%202017) (visited on 04/09/2017).

[83] User-agents. url: https://http://www.user-agents.org%20(access%20date:%20September%204,%202017) (visited on 04/09/2017).

[84] Guido Van Rossum and Fred L Drake Jr. Python tutorial. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[85] Luis Von Ahn, Manuel Blum, Nicholas J Hopper and John Langford. 'CAPTCHA: Using hard AI problems for security'. In: International Conference on the Theory and Applications of Cryptographic Techniques. Springer. 2003, pp. 294–311.

[86] Abraham Wald. 'Sequential tests of statistical hypotheses'. In: The Annals of Mathematical Statistics 16.2 (1945), pp. 117–186.

[87] Chamila Walgampaya and Mehmed Kantardzic. 'Cracking the Smart ClickBot'. In: Proceedings of the 13th IEEE International Symposium on Web Systems Evolution (WSE'11). Sept. 2011, pp. 125–134. doi: 10.1109/WSE.2011.6081830.

[88] Zhengzheng Xing, Jian Pei and Eamonn Keogh. 'A brief survey on sequence classification'. In: ACM Sigkdd Explorations Newsletter 12.1 (2010), pp. 40–48.

[89] Zhengzheng Xing, Jian Pei and S Yu Philip. 'Early classification on time series'. In: Knowledge and information systems 31.1 (2012), pp. 105–127.

[90] Zhengzheng Xing, Jian Pei and Philip S. Yu. Early Prediction on Time Series: A Nearest Neighbor Approach. 2009.

[91] Zhengzheng Xing, Jian Pei, Philip S Yu and Ke Wang. 'Extracting interpretable features for early classification on time series'. In: Proceedings of the 2011 SIAM International Conference on Data Mining. SIAM. 2011, pp. 247–258.

[92] Kiyoung Yang and Cyrus Shahabi. 'A PCA-based similarity measure for multivariate time series'. In: Proceedings of the 2nd ACM international workshop on Multimedia databases. ACM. 2004, pp. 65–74.

[93] Mahdieh Zabihi, Majid V. Jahan and Javad Hamidzadeh. 'A density based clustering approach to distinguish between Web robot and human requests to a Web server'. In: The ISC International Journal of Information Security 6.1 (Jan. 2014), pp. 77–89.

[94] Mahdieh Zabihimayvan, Reza Sadeghi, H. Nathan Rude and Derek Doran. 'A Soft Computing Approach for Benign and Malicious Web Robot Detection'. In: Expert Syst. Appl. 87 (Nov. 2017), pp. 129–140. issn: 0957-4174. doi: 10.1016/j.eswa.2017.06.004.

[95] Igal Zeifman. Bot Traffic Report 2016. Tech. rep. Imperva Incapsula, 2017. url: https://www.incapsula.com/blog/bot-traffic-report-2016.html.

[96] G Peter Zhang. 'Time series forecasting using a hybrid ARIMA and neural network model'. In: Neurocomputing 50 (2003), pp. 159–175.

[97] Linfeng Zhang and Yong Guan. 'Detecting Click Fraud in Pay-Per-Click Streams of Online Advertising Networks'. In: Proceedings of the 28th International Conference on Distributed Computing Systems. June 2008, pp. 77–84. doi: 10.1109/ICDCS.2008.98.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Genova, Italy
December 2019

Alberto Cabri